A LEARNING ALGORITHM FOR ROBOTIC ASSEMBLY

by

PRAVEEN BHATIA

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JANUARY, 1988

A LEARNING ALGORITHM FOR ROBOTIC ASSEMBLY

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

PRAVEEN BHATIA

to the

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JANUARY, 1988

ACKNOWLEDGEMENTS

I wish to express my gratitude to Dr.H.Karnick and Dr.A.Ghosh for their inspiring guidance throughout this project.

I would also like to thank all my friends for their valuable suggestions and kind co-operation.

PRAVEEN BHATIA

TABLE OF CONTENTS

LIST OF FIGURES

Probabilistic Learning System

Praveen Bhatia

Indian Institute of Technology Kanpur

ABSTRACT

This thesis describes a "Probabilistic Learning System" (PLS) which uses its past experience to solve the given problem in a better manner in the future. The program has been mainly written in LISP. Some of the routines have been written in C. The learner (i.e the PLS) develops a Knowledge base from the past experience. This knowledge base is applied in the future for decision taking and solving the problem. The task on which the learner is tested is the insertion of a peg in the hole by a Robot in a mechanical assembly operation. The position of the hole has some probabilistic distribution and the Robot is expected to develop an optimal search strategy and, as its experience grows , to reduce the average distance it travels to find the hole position. Throughout the work attempt has been made to parallel the learning of a human being performing a similar task.

1. Introduction

Industrial Automation has for quite some time been one of the major topics of research. Attempts are being made to develop systems which can run with the least intervention from human beings. In a manufacturing situation (especially flexible automation) the " assembly operation " has proved to be one of the most difficult to automate.

Of late , considerable interest has been generated in the application of Artificial Intelligence techniques to the manufacturing situation. The technological advances of the past decades have brought in challenging prospects of automating and mechanising a wide variety of traditionally human intellectual tasks. One of the remarkable abilities of intelligent human beings , learning , has for long been an active research area.

Our work is an attempt to explore the applicability of a learning system in a manufacturing situation.

Usually a number of basic operations go up to make the whole assembly operation. Peg insertion is one of the most basic operations performed during the assembly phase. We have picked up this problem to test our learning system.

2. Need for a probabilistic learning system :

A learning algorithm has the potential to improve its performance as its experience increases. In an idustrial situation it is normally quite difficult to make a set-up such that the production system always performs near the optimum condition. Also , if the situation is dynamically

changing then it will be next to impossible to keep changing the set-up to meet the near optimality condition. However , if the system itself is capable of inducing some information from its attempts to perform the task, then it is likely that the condition for near optimality can be met by the system dynamically. As the system will be expected to learn from data which may not be complete it is likely that it will have to do some sort of probabilistic induction on the data available to it.

In a mechanical assembly , typically , operations like picking up objects , or mating of two parts , or placing an object in some place are done. In other words most of the operations in a mecahnical assembly involve some sort of interaction between two or more objects. The locations at which these objects appear are determined by some probabilistic distribution. One of the difficulties during assembly is precisely this uncertainity in location of the objects .

Hence the motivation to build the Probabilistic Learner System. We believe that if our learning system can be made powerful it can prove to be a very important tool in the mechanical assembly operation.

Learning has generally been classified into five categories. They are as follows :

3. Types of learning :

[1] Rote Learning : This is the simplest kind of learning and requires no inference of knowledge on part of the task performer. Everything that the system has to do is

programmed explicitly by the programmer. Thus most of the burden for the performance of the system lies on the programmer.

[2] Learning by being told : In this class of learning , the knowledge from the source , which may be rules and some advice , is transformed by the learner into an internally usable representation. The key issue in this learning is the development of procedures to implement advice that is not directly executable by the learner.

[3] Learning by analogy : This type of learning is to transform and apply former knowledge to a new , similar situation.

[4] Learning from examples : The learner is presented with both positive and negative examples of a concept. The learner induces a general concept from the given sets of examples and counterexamples. The concept induced by the learner is such that it just accepts all the positive examples and rejects all the negative examples.

[5] Learning by observation : In this kind of learning , the example events do not contain class information and the learner is, to a very large extent, unsupervised.

4. Overview of the thesis :

The research described here deals with what is called 'incremental learning'. In this type of learning the learner normally has a goal task to perform. The task may be to reach some final configuration given any initial configuration or, as in our case , to insert the peg in the hole

where the hole may appear with some probabilistic distribution unknown to the learner initially. It is important that whether the learner has learnt completely or not , it has to perform the task. It may also so happen that after it has learnt a particular distribution a new distribution may begin to emerge. The learner must have the potential to learn the new distribution (and if necessary discard a part or the whole of the information learnt earlier).

Currently, the search for the hole in a peg insertion problem is done by programming a nominal position for the hole , and then each time searching around the hole exhaustively in a square spiral(with increments of 1 BRU for each round. A BRU is the basic resolution unit of the robot).

This is an exhaustive search method and it potentially has many disadvantages. For example , if the programmer has programmed a position very far from the actual centroid position of the determining distribution ,then the robot is going to travel unnecessarily large distances each time it performs the peg insertion. On the other hand if the workpiece is going to appear from two or more different samples of production runs with (perhaps) different distributions ,then it is quite likely that these two distributions may be shifted , or rotated , or may even be totally different.

Thus even if the programmer has correctly programmed the nominal position near the centroid , the hole may rarerly or even never appear at the nominal position. Even if the workpiece appears only from a single sample and has only one distribution attributed to it , it is quite possible

that the distribution may not be symmetric about the axis about which the square spiral is symmetric. Also as time passes , wear and tear can cause a shift in the distribution. The programmed position may begin to considerably differ from the new centroid.

In other words , the method presently employed in the industry will work well only for those situations in which the overall distribution is static with respect to the space , is symmetric about the axis of symmetry of the spiral , and the probability of finding the hole decreases radially outwards in all the directions.

Clearly , if the robot could learn any of the distributions that it encounters on line , all these inefficiencies could be removed. Moreover learning could very well be profitably used in many other applications in which learning of an unknown distribution is important for its efficient performance.

This is perhaps only one of the many learning situations that a general learning system will encounter in a manufacturing system. So it is very much undesirable to develop an algorithm which can only solve one class of problems. As one of the main ideas of our research work was to be able to get an insight into a more sophisticated learning system , it has throughout been our attempt to look at the higher goal of developing a general learning system which may learn by any of the five methods mentioned earlier.

Consequently , we see our entire work in the light of decision making and we see our entire problem as a product

of some higher level decision making i.e the very decision that this situation has to be solved the way our learner is doing it, is the result of some higher level decision making.

The ability to make decisions is central to AI systems. Expert systems make decisions between alternative options. Programs for "planning" and "design" assemble sets of candidate plan steps or design alternatives , and then decide among the alternatives. Decisions are made in adopting or eliminating hypotheses , in selecting strategies for problem solving, choosing moves in a game and in growing and pruning search trees in learning and discovery. Indeed every heuristic , and every inference in an AI system is ultimately concerned with a decision to follow some path in preference to others.

Our learning system is called the "Probabilistic Learning System" (PLS) and as the name implies , it probabilistically infers knowledge from the data available to it. Though statistical data is used and classified , it differs markedly from the statistical methods generally connoted by the term. The distinction between the methods of "machine learning" and " statistical data analysis " is primarily due to the differences in the way techniques of each type represent data and structure within data. That is , methods of machine learning are strongly biased towards symbolic (as opposed to numeric) data representation.

2.PROBABILISTIC LEARNING SYSTEM - PLS

PLS has been implemented on a 68020 based machine , using the 4.2 UNIX OS. The code has been written in "Franz Lisp".

1. Introduction

We call our learning system the 'Explorer'. The design of the Explorer is based on the following model of the peg insertion task.

Assume :

There are several bins kept in the form of a rectangular grid on a plane ground. A man has to pick up an object from a house nearby and place in the only bin (hole) which has no object inside it. At any time there is exactly one bin which has no object in it. The man has to find the hole without travelling more than some fixed distance ,failing which the search is aborted. After the man has found the hole (or has aborted the search due to limit on the distance traveled) a new situation is set-up and the man has to repeat the task of picking up another object and placing it in a new empty bin . Furthermore , all bins are closed and the man has to open up the bin to test whether an object is present in the bin or not.

Now the problem is to determine the path the man will follow in searching for the empty bin as his experience grows. If the empty bin appears at totally random locations then there is nothing to learn and probably the man will make an exhaustive search for the bin. However, if some

pattern begins to emerge as his experience grows then it is likely that he will modify his search path in future attempts ; but how should he modify his search path so that he can reduce the average distance traveled ?

In the first attempt he has no previous experience and thus will have to follow some strategy which is exhaustive. Let's say , he starts at the center of the grid and goes outwards in a square spiral. After finding the first bin , in his next attempt he spirals around the first solution. Perhaps, to play safe ,in his first few attempts he will traverse a square spiral around the approximate centroid of the previous experience.

When he first begins to sense the formation of a cluster in some region he starts forming a belief from his experience to date. This is the region that he is likely to visit first in his next attempt , failing which he extrapolates his belief to search for the empty bin. If this strategy also fails then he reverts back to his earlier strategy of searching exhaustively by moving around the approximate centroid (without visiting the previously visited bins).

Now let us look at the problem more formally :

A Robot has to perform a task over some event space. In the case of peg-insertion , picking up the peg from a bin , finding the hole and inserting the peg in the hole becomes the task. The event space is the set of all the events that can take place and in peg-insertion it becomes the set of all the co-ordinates where the hole might appear. We also impose the restriction that the hole can appear only at

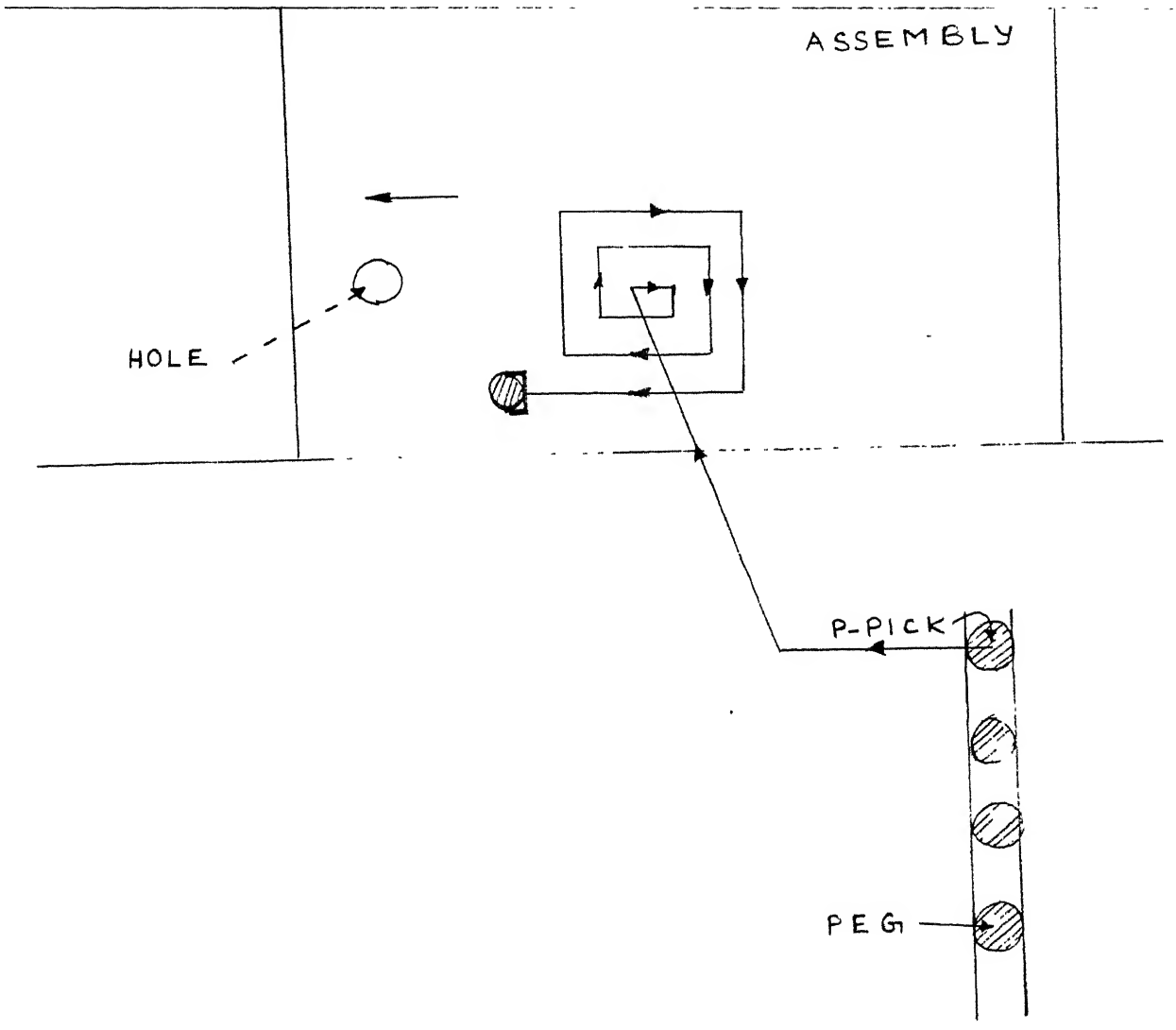


Fig. 1

Schematic diagram for the formulation of the problem.

points with integer co-ordinates . In general, the events are restricted to an integral grid in some abstract co-ordinate space and the distance is defined by the Eucledean metric over these abstract co-ordinates.

2. Formulation of the problem :

To test our Explorer we have formulated the problem of peg insertion as follows :

All the pegs are perfectly oriented and placed in some place "P-pick". The robot has to always pick up a peg from the same spot and insert it in the hole. The peg is always assumed to be present at the same spot for the picking up operation and is perfectly oriented. After picking up the peg the robot has to insert the peg in the hole. The position of the hole is not known. It is in this phase that we are testing our learning algorithm. It is assumed that after the robot has found the hole , it is able to insert it in the hole by the standard methods used in the industry.

It must be noted here that in the whole process of peg-insertion there may be many kinds of information the Explorer can learn to improve its performance. For example , it may be able to learn the distributions of the clearances between the peg and the hole , or it may be able to learn the best angle of insertion etc. Presently we have restricted ourselves to running our algorithm to the stage of searching for the hole.

3. General discussion :

Any learning system minimally comprises of :

[1] An algorithm (the solver) which encodes the strategy for the search task and uses the representation which is acquired and modified as learning progresses.

[2] The learning algorithm which modifies and improves the representation according to the observed behaviour of the system . The figure shows the various blocks and the nature of information flow between the blocks. In this diagram the learning element retains a condensed version of its experience from one iteration to the next.

In the case of peg-insertion the average distance traveled by the robot to find the hole is to be minimized.

Let us take a small example of two cells to illustrate how the strategy adopted by the robot affects the final distance traveled.

Suppose we have just two cells C-1 and C-2 such that to traverse both the cells completely , distances A-1 and A-2 respectively have to be traveled , and let the probabilities of finding the hole in these cells be p1 and p2. (Within a cell the probability is assumed to be uniformly distributed.) Also assume that during the search C-1 is visited first. If the hole is not found in C-1 then C-2 is visited. If the hole is not found in C-2 then the search is aborted and another problem is begun. Thus over a large number of trials, and ignoring the distance to be traveled between the cells, the average distance that the robot will travel will be

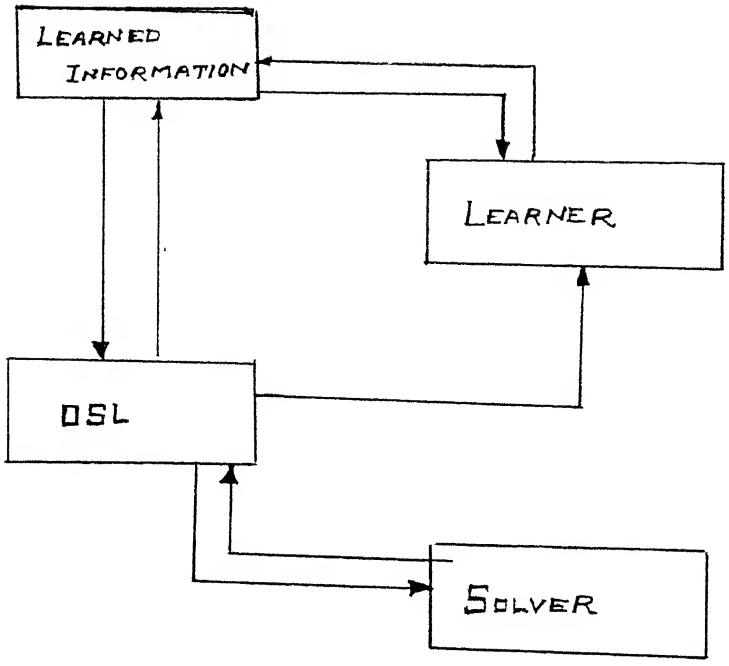


Fig. 2

Explorer has three main blocks:

[1] OSL - The Operating System

[2] SOLVER

[3] LEARNER

'Solver' and 'learner' are unaware of each other's existence. They become active only on being requested by the OSL. OSL interacts with the SOLVER while solving the problem. After he problem is solved the OSL invokes the learner only if it feels the need for learning again. Otherwise the OSL makes only a small modification in the learned information. If the learner is invoked by the OSL then it overwrites the earlier learnt information by the newly learnt information.

$$= p1 * A\text{-}1 /2 + (1 - p1) * A\text{-}1 + (1 - p1)[\, p2*A\text{-}2 /2 + (1 - p2)*A\text{-}2]$$

or we have

$$av\text{-}dis = (1 - p1/2)A\text{-}1 + (1 - p1)(1 - p2/2)A\text{-}2$$

Now generalizing the above case for more than two cells and aborting the search only after the distance traveled exceeds a certain limit A-limit, we have the following analysis :

Let 'n' cells be already formed which have to be visited and let them be numbered C-1, C-2, C-3 ... ,C-n such that the Explorer travels first to C-1 , then to C-2 and so on. Also , let A-1 , A-2 .... ,A-n be the maximum distances that the robot may have to travel to test each cell completely for the presence of the hole. (Note : the distance required to reach the cell is being ignored in the analysis.)

Let the probabilities that the hole is located in these cells be p1 , p2 ,p3 , ... , pn respectively.

Using this strategy the average distance traveled over a large sample of trials will be :

$$= (1 - p1/2)A\text{-}1 + (1 - p2/2)A\text{-}2 * (1 - p1) + (1 - (p1 + p2))(1 - p3/2)A\text{-}3 + \ldots\ldots \text{ till the last cell.}$$

Representing (1 - pi/2)A-i by B-i and assuming that the hole will be found within the A-limit we can represent the equation for the average distance as :

```
av-dis = B-1 + (1 - p1)B-2 + (1 - [p1 + p2])B-3 + .....
+ (1 - [ p1 + p2 ....p{n-1} ]) B-n
+ q*q/2(A-limit - (A-1 + A-2 + A-3 ....+ A-n))
       - B-1 + B-2 + ..... B-n
         - ([p1] B-2)
         - ([p1 + p2] B-3)
         - ([p1 + p2 + p3] B-4)
         .
         .
         .
         .
         - ([p1 + p2 + p3 ..... p{n-1}] B-n)
         + q*q/2(A-limit - (A-1 + A-2 + A-3 .... + A-n))
```

where q is the probability that the hole will not be found within the 'n' cells.

The above analysis assumes that whatever probability distribution of events can be inductively inferred from the history of actual events does indeed match the actual probability distribution (also called the determining distribution) . In such a case we say that the complete knowledge *K is known to the Explorer.

Moreover , for optimality , it is also assumed in the analysis that the ideal inductive inference of the previous paragraph has actually been made ; i.e , the set of past events has been partitioned into cells in an optimal manner and the optimal search order on these cells has been chosen.

The above situation is unrealistic as the Explorer normally will have only partial information and its information is likely to have considerable amount of uncertainity. Also the optimum clusters will have to be decided by it rather than given to it.

Furthermore , the Explorer will be working under a time constraint. In an Industrial situation it will be expected

to perform not only reliably but also quickly. Clearly to find the best possible clustering at any learning stage will be computationally extremely costly and in most cases unnecessary.

Thus to reduce the time complexity of the algorithm , we have restricted the cells to rectangular cells only. Also , the algorithm attempts to find a good clustering rather than the best possible clustering.

Our clustering algorithm takes the set of all events and partitions it according to some set of rules provided to it. First , the clustering algorithm partitions the event space into cells , and then uses these cells to form higher level concepts (i.e clusters) to form what we call the set of beliefs "B".

Please see the figures 3, 4 & 5 which show in the case of peg-insertion problem

1. The event space

2. The cells , and

3. The clusters

4. Discussion of the Algorithm :

We will discuss the whole algorithm for the PLS by describing the various procedures that form the main frame-work of the PLS. Simultaneously we will also keep describing how the various procedures are linked with each other in the PLS.

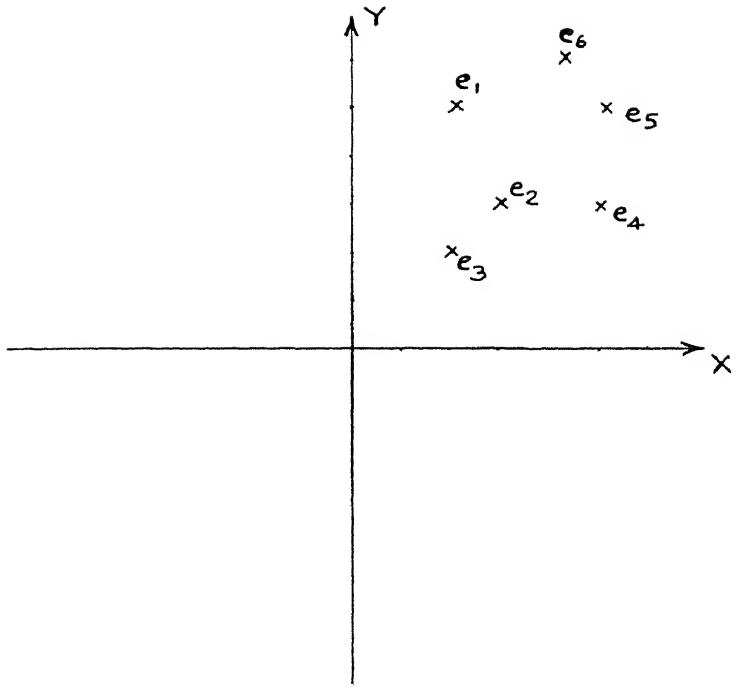


Fig. 3

The event space 'E' for the search in a peg-insertion problem is the x-y plane. It consists of various events e1,e2,e3,...en which are essentially the x and y co-ordinates for various positions where the hole has occurred.

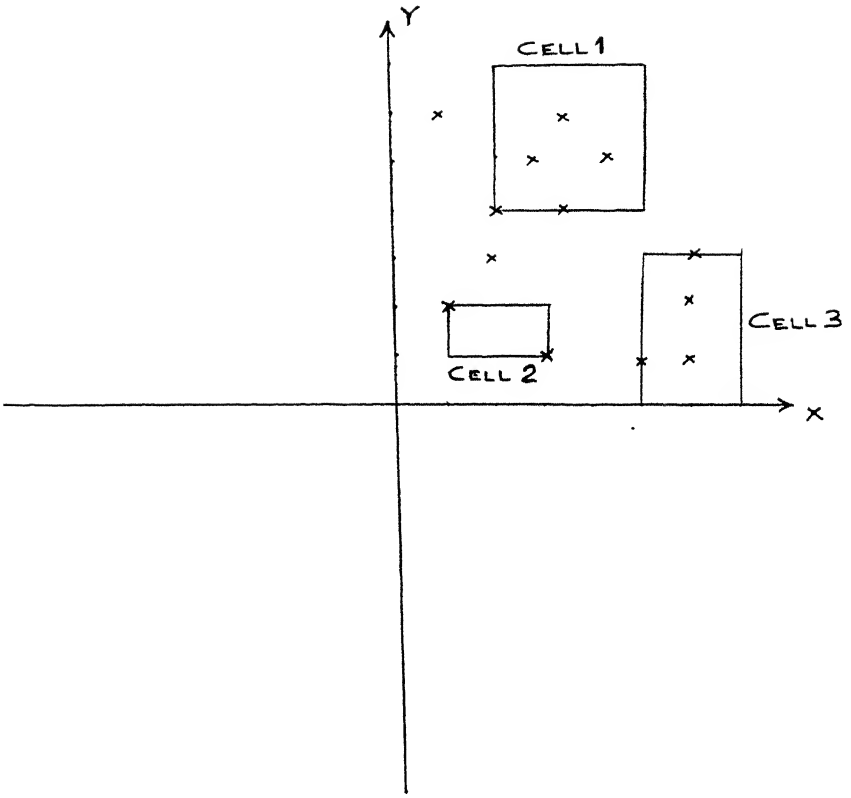


Fig. 4

A cell is a cluster of some events in the event space 'E'. These cells represent the concrete belief of the Explorer denoted by 'C-BELIEF'.

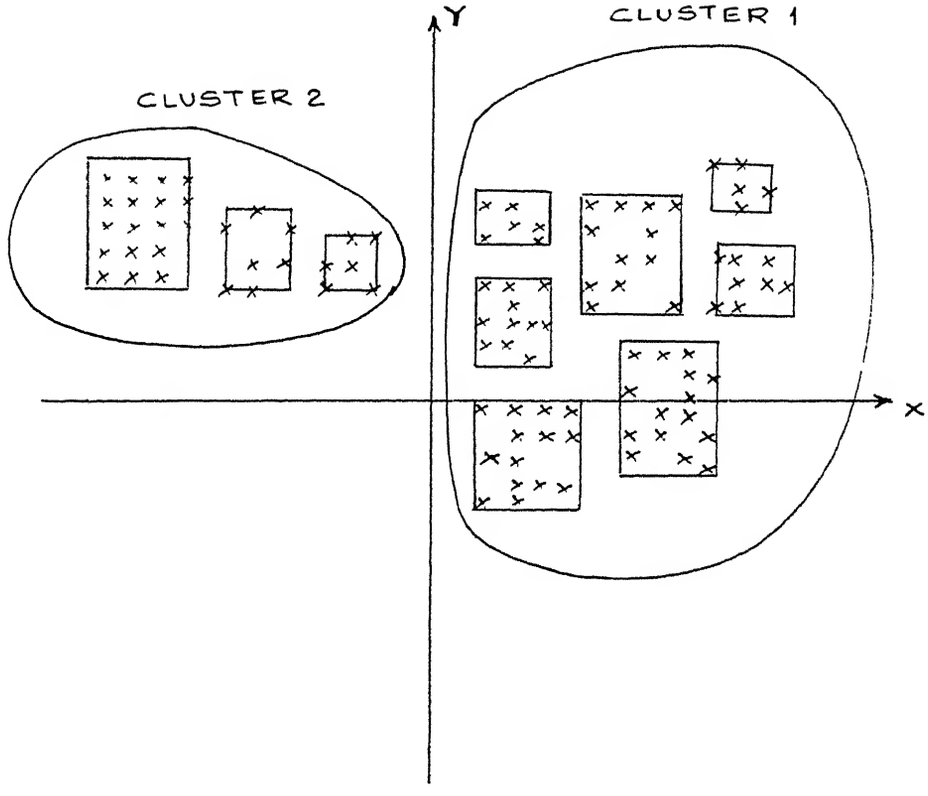


Fig. 5

A cluster is a group of cells which are bound by some set of rules. Each cluster is extrapolated to acquire some information. The set of these informations is the extrapolated belief of the Explorer represented by E-BELIEF.

4.1. Basic Concepts :

Given any set of events ,we wish to cluster them so as to be able to make decisions (for search). The main function of the clustering algorithm is to collect information in such a manner that it's output is

1) easy to use

2) fast to access and is

3) representative of a very large percentage of the knowledge domain.

The Explorer stores most of the information in the form of cells. These cells are the quanta of information over which higher level clusters are formed by the Explorer. Any information that the Explorer extrapolates is from these higher level clusters.

4.2. Clustering algorithm

The aim of clustering data is to be able to finally use the clusters for some task. Thus the nature of clustering will be considerably influenced by the final task to be performed.

The clustering algorithm is typically supplied with the following information :

[a] What are the minimum requirements for the formation of a cell ?

[b] What conditions must be satisfied by the elements of the cell ?

[c] What criteria are sufficient to split a cell into smaller cells ?

In the case of peg-insertion the following information is provided to the algorithm for partitioning the event space .

[1] A cell must represent at least 'n' % (user defined) of the total number of instances which occurred in the event space.

[2] Within a cell a minimum number of events must be neighbours i.e adjacent to each other.

[3] A cell must be divided if the Explorer is confident that on division of the cells less amount of resource is spent in testing the new cells than when the cells stay undivided.

Essentially the algorithm minimizes the differences within the cells and maximizes the differences between cells.

4.2.1. Partitioning Algorithm :

The partitioning algorithm begins by fitting a large cell in the event space such that all the events are covered. Now it invokes a procedure " divide " which given any cell divides it into as many smaller cells as is possible , consistent with the rules given for partitioning (as has already been described earlier).

After the partitioning is complete , we essentially have a number of small cells. Quite possibly some of the

neighbours of the cells which deserve to be part of the cell could have been left out while dividing . Thus after the event space has been partitioned the "grow" procedure is invoked which essentially picks up the neighbours of the cell subject to the condition that these inclusions improve the rating of the cell.

4.2.2. Procedure divide

This procedure takes a cell and divides it in accordance with the rules of partitioning . At any one given time the cell can be represented by a set of tuples of co-ordinates of the event space. For example , in the peg-insertion problem each tuple consists of the X and the Y co-ordinates of an event. At any time the division of the cell takes place only with respect to one co-ordinate .(For the event space with X and Y as the only co-ordinates the cell is divided either parallel to X-axis or the Y-axis). The cells formed from the division of the parent cell are again divided recursively till no more split is possible.

4.2.3. Division of a cell :

A trial plane for splitting the cell is tried at each integer value of the co-ordinates in the event space (for peg-insertion there are only two such co-ordinates ; viz X and Y).

Figures 6 & 7 show these concepts for the case when there are only two co-ordinates in the event space.

Each such division is rated (the function used for

rating is presented a bit later). The division is made at the plane where the rating is maximum . A cell is not divided if the sum of the rating of the sons is less than that of the parent . Also , if on division one of the cells has elements less than 'n' % of the total number of the elements , then the information present in the cell may be lost to the learner because such a cell will not be allowed by the clustering algorithm (condition 1 above) .

Thus , while rating each division a penalty is attached if information is lost in the manner described above .

Some of the information lost due to the discarding of the cell can sometimes again become part of a cell later on when the procedure "grow" is invoked . Procedure "grow" essentially tries to pick up the events in the neighbourhood of the cell.

Following definitions will be useful in describing the rating function.

radius of a cell : r

r = (4 Area-of-cell)/perimeter-of-cell

Influence of a cell : I

I = (probability-of-event-occurring-in-cell)/r

Level-of-confidence : *C Level-of-confidence is representative of how sure the Explorer is that its set of beliefs "B" and the determining distribution do not differ much. We represent level-of-confidence by a fraction '*C' such that Ø <= *C <= 1.Ø

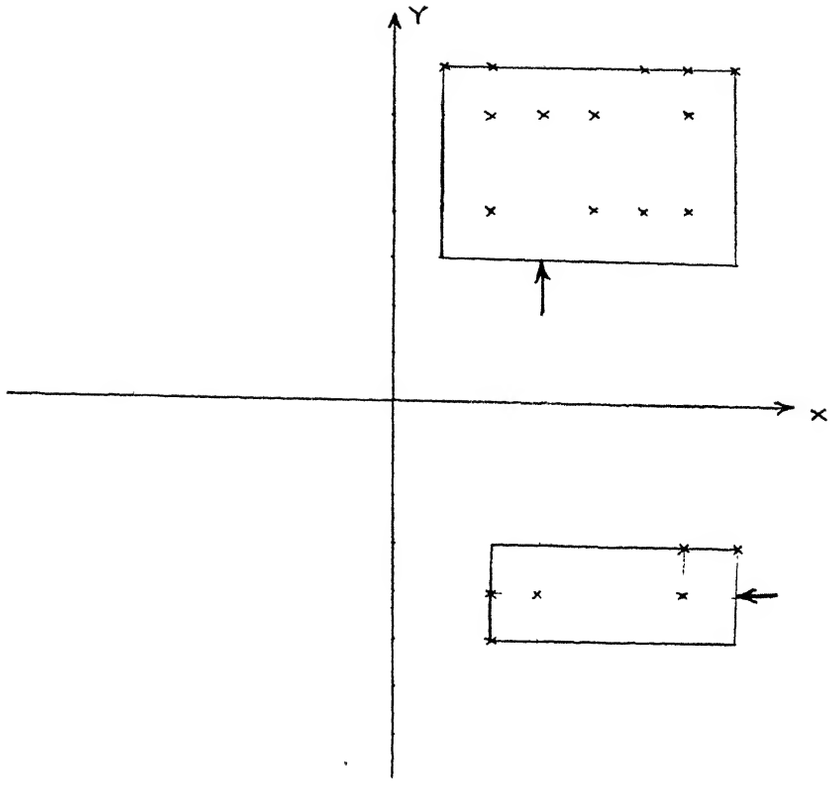


Fig. 6 Division of a cell.

A trial plane for splitting the cell is tried at each integer value of the variables in the feature space. The cell is not physically split now. Each trial division is rated and the best division is made at the plane where the rating is maximum

Result of division of the cell is one of the following:

[a] No division at all

[b] One smaller cell

[c] Two smaller cells

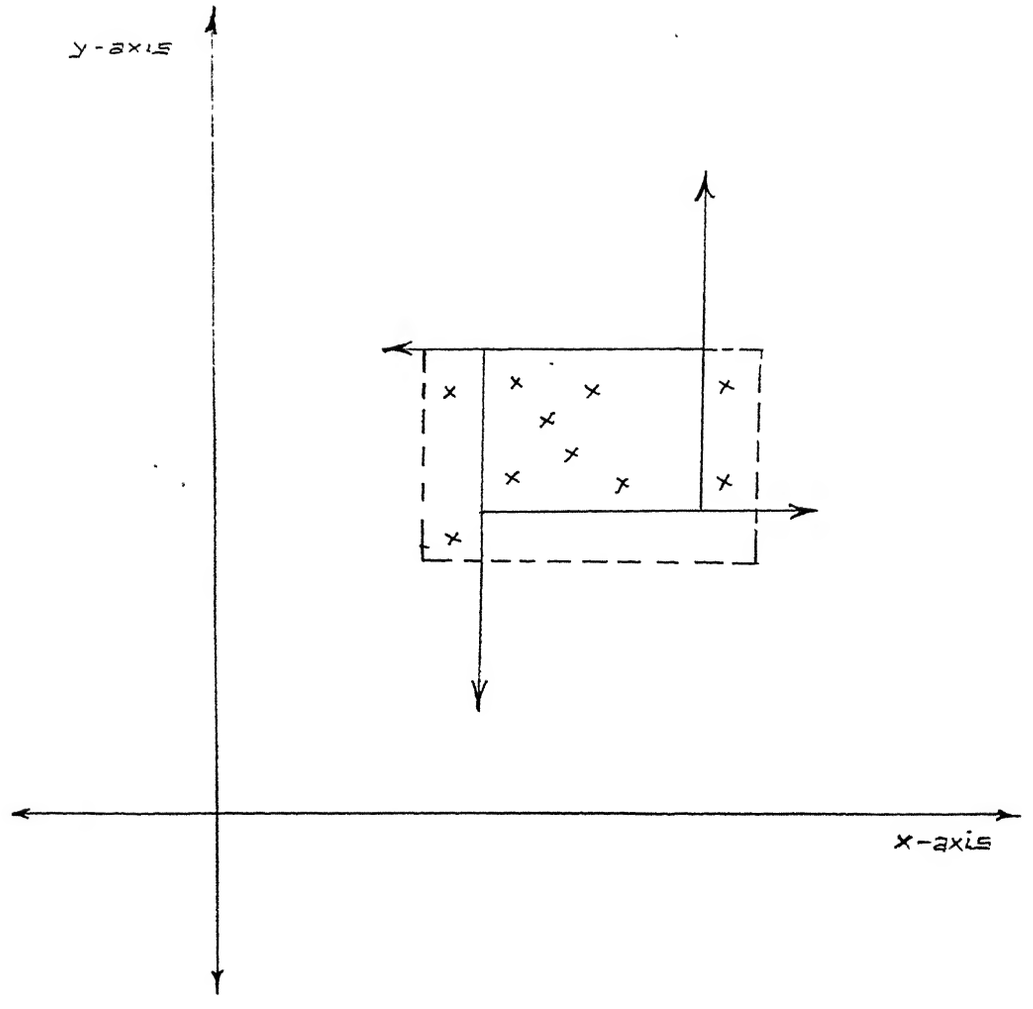


Fig. 7

Procedure 'grow' expands the area of the cell to include more events. Growing is done by expanding each side of the cell one by one.

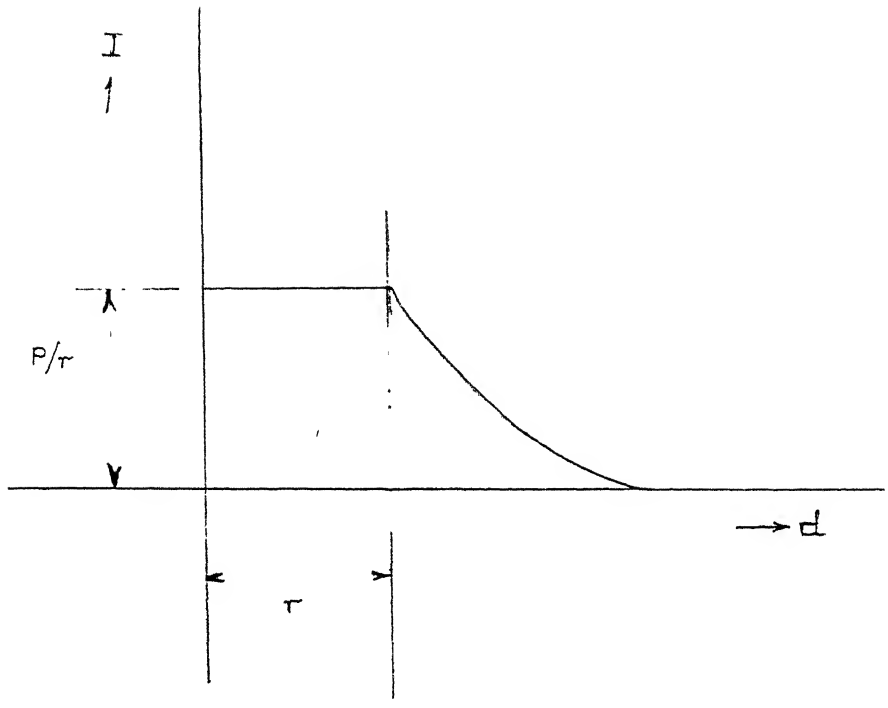


Fig. 8

Influence is very similar to the concept of gravitational field of earth.

Let a cell have a radius 'r' (computed by the formula 4 * Area/Perimeter) and let the probability of success in the cell be 'p'.

Then potential at a distance d is

= p/r for d < r

= p/d for d >= r

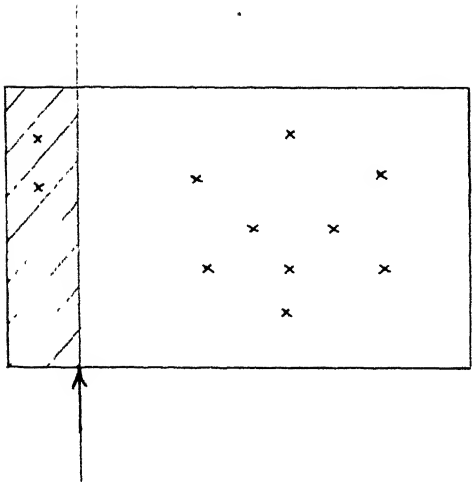


Fig. 9

It may so happen that while trying a trial plane, one of the cells may not satisfy the minimum requirements for the formation of a cell. Obviously, if physically division is done at this plane, then one of the cells will be discarded. The information carried by the discarded cell is thus lost. While trying the trial planes.rating of such a division includes a penalty proportional to the lost information.

In the peg-insertion problem , level-of-confidence is the probability that the Explorer will find the hole within the cells.

Let us take a cell 'C' and if this were to become a part of the belief then let the resource that is expended on an average if this belief were tested first be 'R' (In the case of peg-insertion the distance traveled by the Explorer to test the possibility of the solution within the cell is the resource 'R'). Let us represent the influence of the cell C as 'I'.Also , at this stage , let the level of confidence of the Explorer be '*C'.

Now if the cell C has to be divided into two cells c1 and c2 , and either of the cells do not meet the requirement for the formation of a cell the particular cell will not be formed i.e the resource expended on that cell will be Ø but it will face a penality for losing the influence of the particular cell in the formation of a set of beliefs .

After partitioning the event space into cells ,the learner starts clustering these cells into higher level structures . This permits it to recognize patterns which cannot be described by the simple cells described above. In particular , cells necessarily are rectangles oriented along the axes with uniform distribution of events assumed inside them. Clusters, on the other hand , will be considered as representing arbitrarily oriented ellipses of arbitrary eccentricity with event probability assumed to be decreasing radially outward from the centroid of the ellipse . For example, if the hole appears as a mixture of two normal distributions (from say 2 samples) , then it will be learned by

the Explorer as a set of two clusters , each of the clusters being a set of closely related cells .

Here , we shall introduce two terms which will be used later on.

[a] Concrete-belief :

This is the set of all the cells which the clusterer has formed. From now on we shall represent this belief by "C- belief".

[b] Extrapolated-belief :

This is set of higher level concepts which the clusterer returns. From now on we shall denote this by "E-belief".

Intuitively , C-belief is just an unordered collection of chunks of information while E-belief is a collection of information which is ordered. In E-belief the ordering is done such that some sort of pattern can be observed in the collection of cells.

Intuitively , the learner must try to maximize its C-belief and E-belief over the determining distribution *P. *P is usually unknown to the learner. It is quite likely that the distribution P that the belief of the Explorer represents is very much different from *P. To form a belief from wrong information is very risky because the likelihood of biases is now very much increased. It will be very much desirable that till the learner is very confident of its learning it does not make any assertions that are formed by rejecting some seemingly unimportant information. Another

way to look at it is that till the learner doesn't have enough confidence in its learning , it sticks on to a large number of small pieces of information however unconnected they might seem. In the extreme case when the learner has zero confidence in its learning , no information is ignored. At the other extreme of supreme confidence small pieces of information are likely to be ignored by the learner.

Explorer implements the above concept by using a very simple heuristic as explained below.

If before division of the cell the resource being expended was R and the influence of the cell was - I and the values after the division were R'' and I'' then one can represent

the unit change in resource expended dR/R as (R - R'')/R , and

the unit change in influence of the cells dI/I as (I - I'')/I.

If the level of confidence is high then (-dR/R) indicates how good it will be to go ahead with the splitting of the cell

and if the level of confidence is low then (dI/I) indicates how bad it will be if the learner forms biases. "Divide" goes ahead with the division if the following expression is negative :

[(dR/R) * level-of-confidence

+ penalty

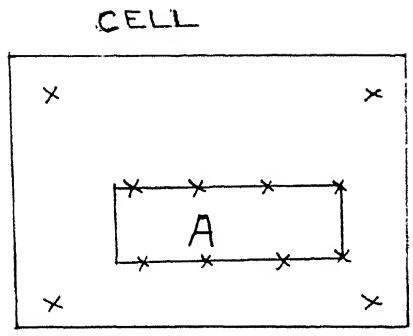


Fig. 10

A negative dR/R has the effect of trying to find the major cluster of events (the region marked as 'A' in the figure) by ignoring the relatively less important events (the four events at the corners of the cell).

A positive dI/I has the effect of trying to prevent the loss of information by ignoring the relatively less important events (the 4 corner points).

When the level-of-confidence of the Explorer is high it tends to give higher weightage to dR/R. However when its level-of-confidence is low it gives higher weightage to dI/I.

+ (dI/I) * (1 -level-of-confidence)]

In the above expression dR/R and dI/I are opposite in nature. While dR/R attempts to ignore smaller informations and form stronger beliefs , dI/I tries to avoid the split of the cells. From another point of view , dR/R tends to look for local patterns while dI/I attempts to look for global pattern.

In the case of peg-insertion

the distance that the robot travels becomes the "resource expended"

"level-of-confidence" is the probability that the Explorer will find the hole within its belief

"influence" of a cell is the potential developed by the cell around its immediate periphery (calculated by the ratio of the probability of finding the hole within the cell and the radius of the cell)

and ,

"penalty" is the number of events that have occurred within the cell in the past (in the case of peg-insertion the event space is the X-Y plane).

C-belief are stored in the form of rectangular cells giving them some properties typically characterizing the cell.

However , E-beliefs are stored differently.

First let us discuss how the E-beliefs are formed.

4.3 Formation of E-beliefs :

The learner picks up all the cells in a list and sorts them based on some heuristic. In peg-insertion we have used the density of the cells for sorting the cells. The first cell in the sorted list is picked up. Now the neighbours of this are picked up from the remaining cells. A cell 'c1' is neighbour of cell 'c2' only if the density of c1 <= density of c2 , percentage change in density is less than some m% and the gap between the cell is within u% of the distance between the centroids of the cells.

If for the two cells r1 and r2 are the radii (as defined earlier),and d is the distance between the centroids of the cells , then

gap is

= d - (r1 + r2)

After the first cell has picked up all its neighbours , the neighbours that have been picked up by the cell , pick up their neighbours. Finally a stage is reached when the cells are unable to pick up any more cells. Now this set of picked up cells forms a higher level of cluster. To get more clusters the above mentioned procedure is repeated with the current first cell in the remaining cells.

The result is a collection of higher level clusters.

These clusters are now to be stored and used efficiently.

The treatment of these clusters will vary depending on

what concept they are representing and how they are going to be used for decision making.

In peg-insertion these clusters approximately represent the distributions learned by the learner. To store these clusters in a condensed form ,the following transformation is done on each of the clusters.

[1] find the centroid of the cluster

[2] shift the X-Y axes to this centroid

[3] calculate the second moment of inertia about these axes

[4] calculate the product of inertia

[5] calculate the rotation "alpha" required for orienting the axis in the principle directions.

[6] calculate the radius of gyration around the new X and Y axis as

a = sqrt (Ixx-new / A-total)

b = sqrt (Iyy-new / A-total)

[7] calculate the probability of finding the solution in the cluster "p"

Thus , each cluster is characterized by 5 quantities (p ,alpha, a ,b, centroid) which we may call a "belief". These quantities are used by the OSL , during the actual phase of searching the hole. A "belief" is chosen and its environment is set by a procedure called the "set-env". The environment set by the "set-env" is normally an elliptical field. The solver blindly tries to move towards the maximum

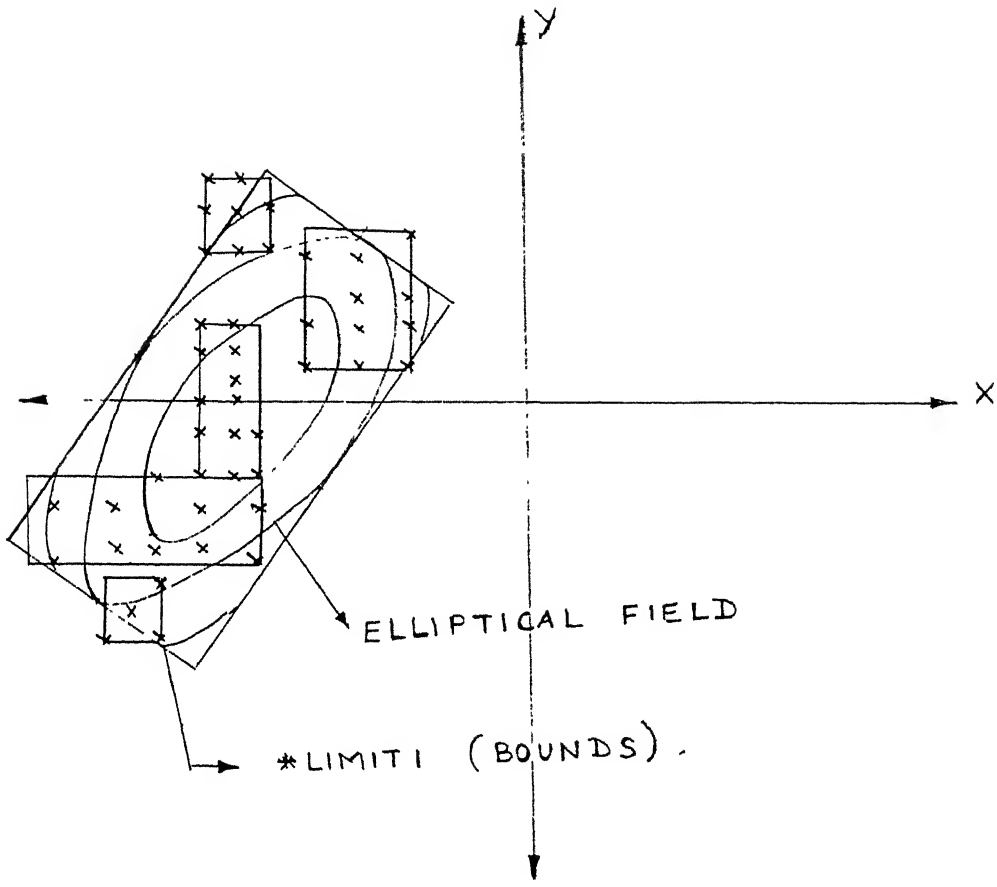


Fig. 11

Potential fields set up by the OSL for the SOLVER

potential without visiting the previously visited points. This scheme allows the OSL to gain control over the solver and thus enforce any decision taken by the OSL on the solver.

Now let us continue with our discussion of the cells.

For any new information to be a candidate for belonging to a cell or forming a cell of its own , the particular information must satisfy some conditions. Some of the common conditions are that it must at least contribute some minimum amount to the whole of knowledge. Or, in other words, this information must have some minimum likelihood of being important in the decision making process. Any information that adds to the already existing cells is used to increase the confidence that the Explorer has in the utility of the particular cell. Thus the likelihood of using the particular cell has been increased for decision making (of course only to the extent of the importance of the new information !).

4.4 Discussion on biases :

It is very likely that the Explorer may get biased by its previous success and derive a completely wrong picture of the actual knowledge *K. Unbreaking a bias is normally quite difficult unless the system itself realizes that its performance may be bad because of the biases that it has developed. Presently while searching or learning the Explorer begins to suspect the formation of a bias whenever it begins to perform badly (i.e suddenly most of the solutions are being found very much outside its belief.) or it is

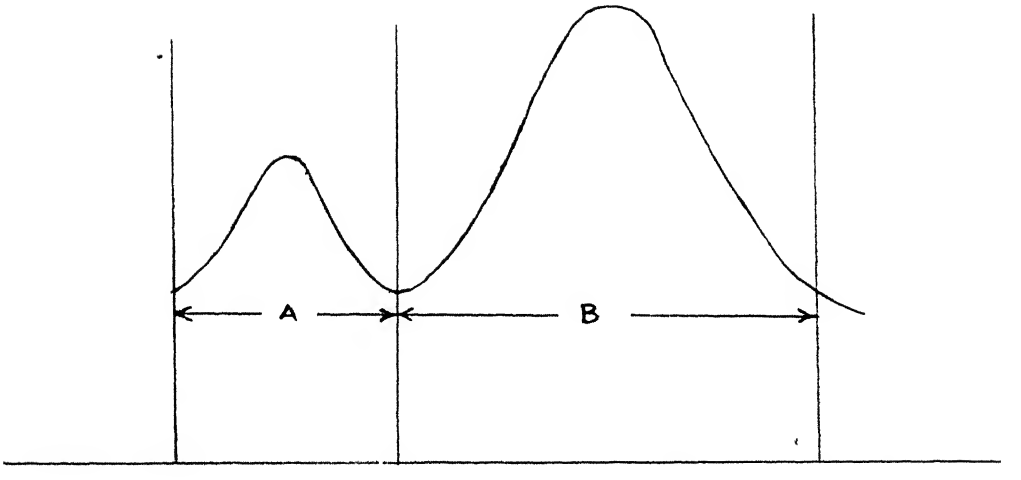


Fig. 12

If the determinining distribution is as shown above, then depending on how the Explorer receives the points , it might form a bias around 'A'. Due to this bias the Explorer may tend to spend its distance resource around 'A'. The Explorer has some simple mechanisms to break this bias.

Biases can very easily be formed with other distribution also, especially when the information available with the Explorer is not representative of the determining distribution.

crossing the self imposed bounds [see footnote].

In such cases the Explorer decides to temporarily forego its learning till more concrete information has been attained.

4.5.1. General discussion on the database of the Explorer :

Let " E " be the event space (e1 e2 e3 e4 ...). Then there exists a probability distribution *P over it and this probability distribution is not known to the Explorer.

The Explorer begins the task by having only the knowledge of the approximate position of the hole that has been programmed by the programmer. For the first few attempts the Explorer moves around the centroid in a square spiral thus searching exhaustively till a reasonable database has been acquired by it on the basis of which it can learn and thus formulate a belief.

Let the actual complete knowledge (not yet known to the Explorer) be represented by ' *K ' implying that if the Explorer had the complete knowledge *K then any optimal strategy developed by the Explorer would be the best possible strategy that it can develop to do the task.

However , the Explorer never has the complete knowledge *K.

At any stage , the Explorer has accumulated some infor-

Explorer has a small operating system implemented in it. It is called the OSL. One of functions of OSL is to set some limits within which any belief will be extrapolated. More will be discussed about the OSL later.

mation from its previous attempts to solve the problem. All the attempts made by the Explorer in solving the problems results in the Explorer accumulating some sort of information.

The information possessed by the Explorer is hardly complete and every move of it has to be based on this incomplete knowledge. (Any probabilistic distribution that describes the experience is possibly the actual knowledge *K.) When the knowledge is complete and precise , then a fixed distribution *P exists and is known completely. On the other hand is the the case of "complete ignorance" when only the bounds of the task are known. In this case any of the possible distributions *X satisfying the bounds is possibly the actual Knowledge *K. In general K is some subset of *K.

4.5.2. Updating the database :

The learning system will also require to update its knowledge as it encounters new experience. It will have to update its inference rule and will have to revise database given the new evidence. This is necessary primarily because we cannot afford to spend computing time to learn the clusters afresh after each iteration. Hence until a need for learning afresh is not felt , only a minor update is done.

Depending upon the new experience updating may have to be done differently. The following two cases can be easily distinguished :

[1] New evidence affects the unknown probability.

This implies that either the Explorer is encountering an event related to the past experience but has not yet been learned , or it is encountering a new situation unrelated to the past (that is a new pattern begins to emerge and the learner might have to unlearn some or all of its previous experience). A single such event is usually not sufficient for the Explorer to invoke its learning to learn afresh . The Explorer considers learning anew again only if some minimum percentage of the total number of events have occurred outside the C-belief. Also ,these events must have some chance of being able to form a new cluster of their own or to be able to extend the existing belief. Explorer presently keeps track of the number of events outside the C-belief which have at least one event adjacent to it. When this total exceeds beyond a certain user defined limit the Explorer decides to learn again. In the later versions of Explorer we hope to implement a better scheme to decide when to learn. A very good scheme would be to recognize the formation of a cell outside the C-belief which has not yet been learned. However , this recognition might result in the use of too much of computing time and for the present we have avoided it.

[2] The new evidence corroborates what is already known till now.

This would result in an increase in the confidence of the Explorer in its own belief. Some minor update may be required in the estimates of its C-belief.

Strictly speaking, any knowledge K must be looked upon as having been derived from some body of evidence 'Z' and

thus denoted by K(Z) , say.

As long as K remains fixed within a problem, there is no need to formalize the dependence of the new experience on Z. If however the new experience is different , and is affecting the confidence of the Explorer in its belief , then this new evidence must be taken into account to modify the knowledge K of the Explorer.

4.6. OSL the operating system for the learner :

Very similar to how an operating system manages the resources in a computer , we have felt that such a concept should prove to be very useful in a general learning system. In Explorer a small operating system has been implemented which we have called the OSL .

OSL becomes operative when the Explorer is unable to find the solution within its C-belief. As soon as the Explorer has checked its C-belief it starts checking its E-belief . E-belief , as we have already indicated earlier , is a set of information derived from the high level clusters and each element of the E-belief now demands being considered first and being considered for as long as possible. Because the Explorer has a limit on the resource it can spend (the distance it can travel to find the hole) these elements of the E-belief must be allocated the resource in the most efficient manner. Also , one of the major tasks of the Explorer is to minimize the use of the resource as much as possible. Hence we see here that the act of considering an element of the E-belief over another , and using it to do the search is equivalent to an operating system allocating a

resource to one among the many requesters. Another very important function of the operating system is to set bounds for an element of E-belief such that the particular belief will not be considered if its use results in the stepping over of the bounds. In the peg-insertion problem the OSL might generate bounds over an element of E-belief as a rectangular area oriented along the principal axis of moment of inertia .

OSL controls both the "learner" and "solver" of the Explorer. The Learner and the Solver are totally unaware of each other's existence and they work independent of each other . At any one particular time only one of them is operative (depending upon the instruction given by the OSL).

On being instructed by the OSL ,the learner learns all the information afresh and overwrites the earlier learned information by the newly learned one.

The solver blindly follows the instructions given by the OSL. The solver understands only two types of instructions :

[a] To check if the solution can be found within a cell .

[b] To follow a potential field set up by the OSL.

OSL in general sets up an elliptical field (which may be displaced and rotated about the co-ordinate axes)

We now discuss the strategy followed by the Explorer.

4.7. Strategy followed by the Explorer

The Explorer at any stage has formed a set of beliefs E-beliefs and the C-beliefs . Its first decision is to check if the solution can be found within the C-belief. If it fails , then it makes note of the fact that the solution has turned out to be outside its C-belief. This information is useful in keeping track of how completely the Explorer has learned the determining distribution, and is useful in simulating the concept of confidence which the human beings display during the learning stages.

Next, the OSL picks up the elements of the E-belief and assigns to each a numerical estimate of the strength of the belief. The OSL now picks up a packet of resource and allocates it to the elements of its E-beliefs - allocating more to the strongest belief and least to weakest belief. Now it considers the belief with the maximum strength and sets its potential field .The solver will blindly follow this potential till the resource allocated to the current belief is exhausted.

The OSL also decides a bound within which the particular belief will be considered to do the search. We have called this limit as "*limit1". The Explorer now does its search based exclusively on this belief until either its resource is exhausted or the extrapolation goes beyond the limit "*limit1" set-up by the OSL. If the bounds are crossed , irrespective of the resource remaining with the particular belief the process of searching for the solution based on the particular belief alone is aborted. The OSL now sets a different potential consistent with all the

beliefs but centered around the current belief being considered. To illustrate this point , let us consider a belief for which a circular field has been set up . Also , let the potential field consistent with the all the beliefs considered together be an elliptical field centered at some point different from the centroid of the current belief.

Now if the bounds are crossed ,the shape of the field that is set up is consistent with that of the whole pattern , but this field is now centered around the centroid of the current belief. The operating system also sets in a global bound for all the beliefs. We have called this as "*limit2" . If this bound is exceeded then the Explorer automatically assumes that it has either developed a bias or has encountered a rare situation. This is when the Explorer temporarily foregoes all its learning and returns to its crude strategy of exhaustive search for the solution. However , if the Explorer re-enters the bounds it resumes its earlier strategy consistent with its learning.

3. RESULTS AND CONCLUSIONS

1. Experiments with the Explorer

We have simulated a number of situations of peg-insertion to test the performance of the Explorer . Its performance is compared with the Square spiral method presently used in the industry . As has been mentioned earlier , the Square spiral method works well for a limited number of cases only . Its performance is best when the programmer has exactly programmed the centroid of the "determining" distribution as the starting point for the search .

To compare the performances of the Explorer and the Square spiral method we used a slightly improved version of the Square spiral method. We always begin the square spiral from the current centroid of the distribution known to the robot .

A situation is simulated by allowing the user to interactively specify the distributions he would like to have in the X and Y axes for the appearance of the hole .

Depending upon the distribution requested by the user , appropriate questions are asked by the program so that it can generate the requested distribution. For example , if a user requests a normal distribution in ~~the~~ one of the axes , then the program asks the user to specify the mean and the variance he would like to have for the normal distribution.

During the simulation , options for performing transformations on the generated distribution are given. This gives the user the flexibility to generate different

combinations of distributions.

After the points for the experiment have been generated , they are fed to the two algorithms

[a] The algorithm which finds the hole by moving in a square spiral around the centroid .

[b] The Explorer

The same points are fed to both the algorithms and the performance of the both the algorithms is compared statistically .

In this thesis we have presented the statistics for six different problems .

The Explorer does not learn the clusters after each iteration . As has been discussed earlier , the Explorer decides to learn afresh only on being instructed by the OSL. Thus the points at which the Explorer decides to learn everything afresh are quite important in depicting the performance of the Explorer .Thus graphs have also been plotted with respect to each learning point . We call this the "current learning point" .Let us call the span between the previous learning point and the current learning point as the "last lap" and the region from the beginning till the current learning point as the "overall lap" To show how the performance of the Explorer improves with each learning we have plotted graphs which give the running average of the two algorithms in the last lap and the overall lap with respect to the current learning point.

The complete statistics for the two algorithms for all

the six problems solved is presented in a tabular form in Appendix A.

The statistics for the two algorithms have also been compared by plotting graphs between the problem number being solved vs the average distance traversed by the two algorithms for three of the problems. These graphs have also been attached to the thesis.

2. Descriptions of the experiments :

Problem no. = 1

In this experiment 200 points were generated by the computer such that in the X-dir the distribution was uniform varying from -5 to +5 ,and in the Y-dir uniform varying from +4 to +6 .

No transformations were applied on the generated data.

Problem no. = 2

In this experiment 150 points were generated and in both X and Y axes uniform distributions varying from 0 to +5 were fed in .

The generated points were transformed by rotating them through 180 degrees about the X and Y axes . The generated points and the transformed points are mixed to give a total of 300 points . This set of 300 points was fed to the two programs.

Problem no. = 3

In this experiment exponential distribution with

expected value of 2 was fed in for both the axes .

Number of points generated were 200 and no transformation was applied .

Problem no = 4

In this experiment 100 points were generated (both the axes had exponential distributions fed in with an expectance of 2). The generated points were transformed by first rotating them about the axis by 180 degrees and then shifting the axis to (5 0) . Thus 200 points were fed in to the algorithms .

Problem no. = 5

Normal distributions in both the axes with "mean" 0 and "variance" 3 were fed in . No transformations were done and the number of points generated was 239 .

Problem no. = 6

Normal distribution in both the axes .

X-axis has a mean 4 and variance 2 , while

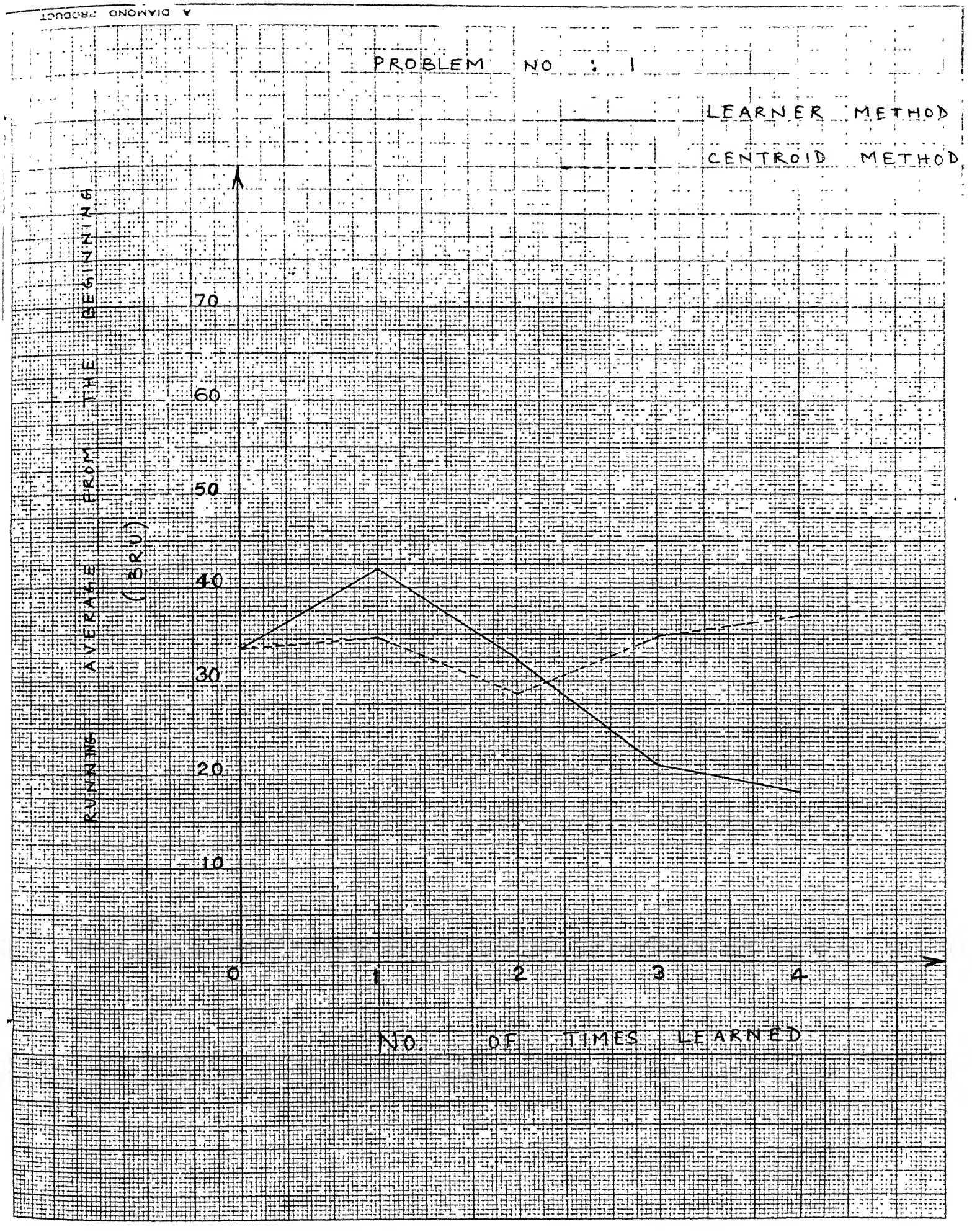
Y-axis has a mean 0 and variance 2 .

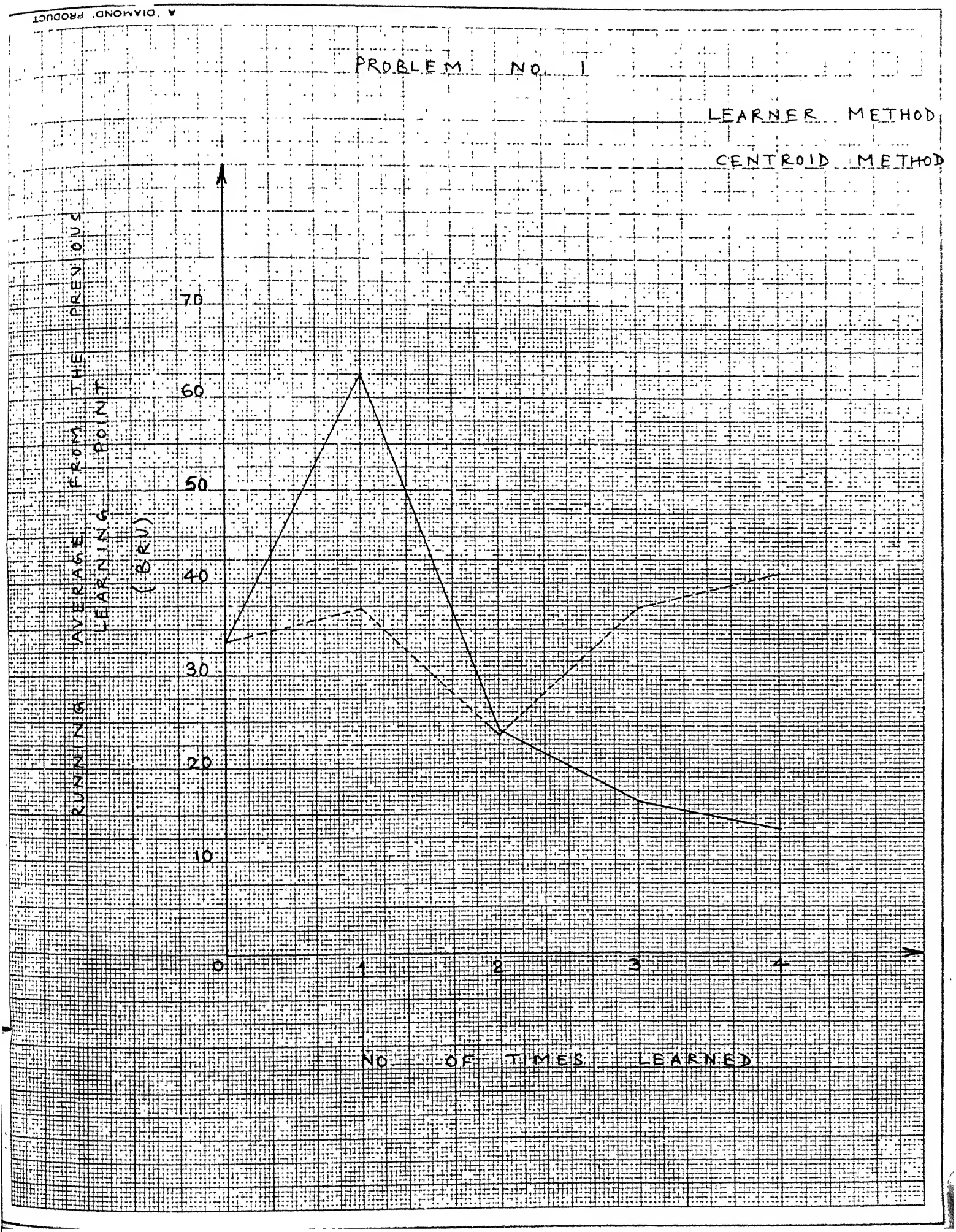
The generated distribution was rotated by 180 degrees .

These two were mixed to form a distribution which is a combination of two normal distributions.

3. Results and Discussion :

It has been seen throughout that the performance of the Explorer has been better than the centroid method . In

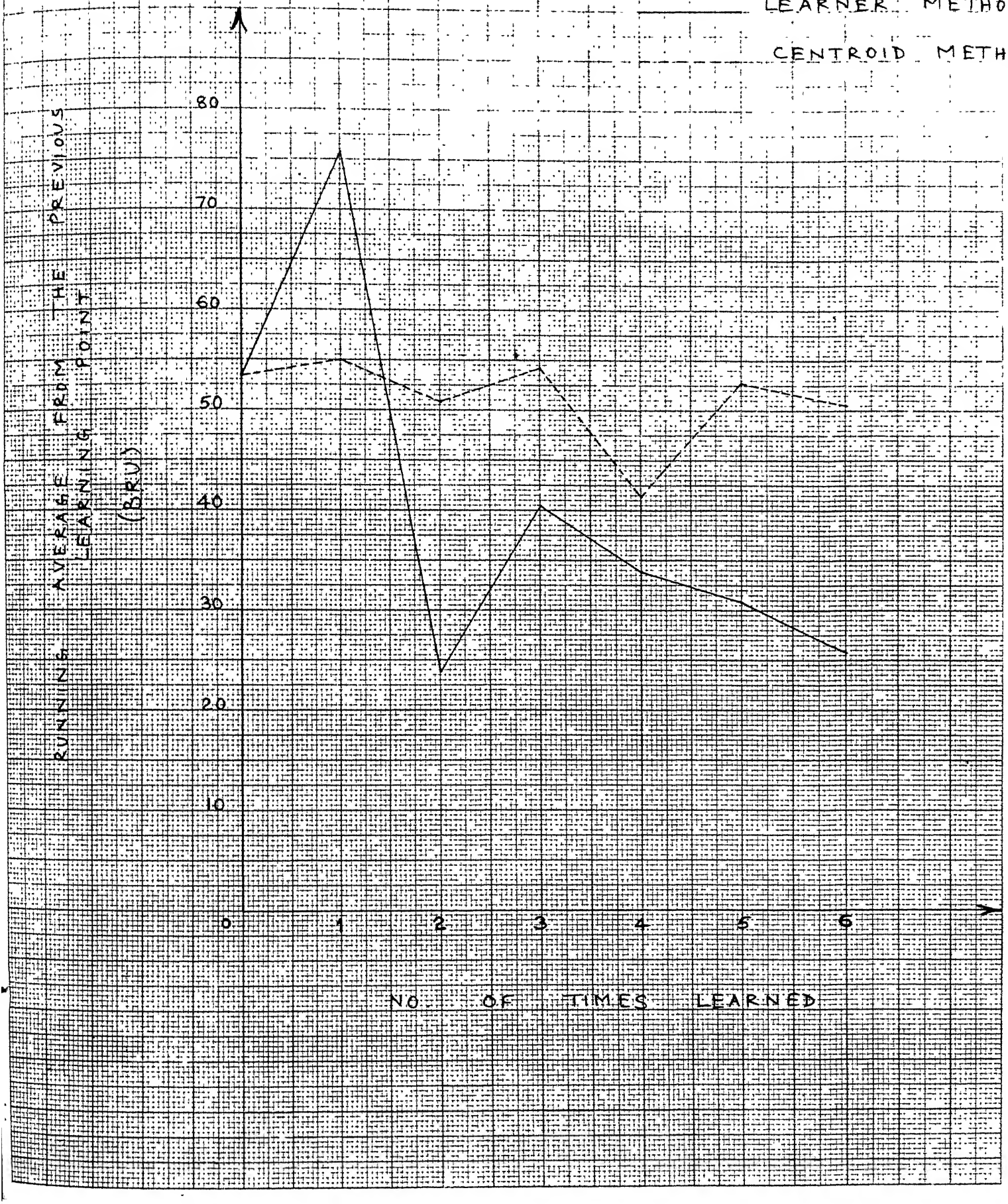


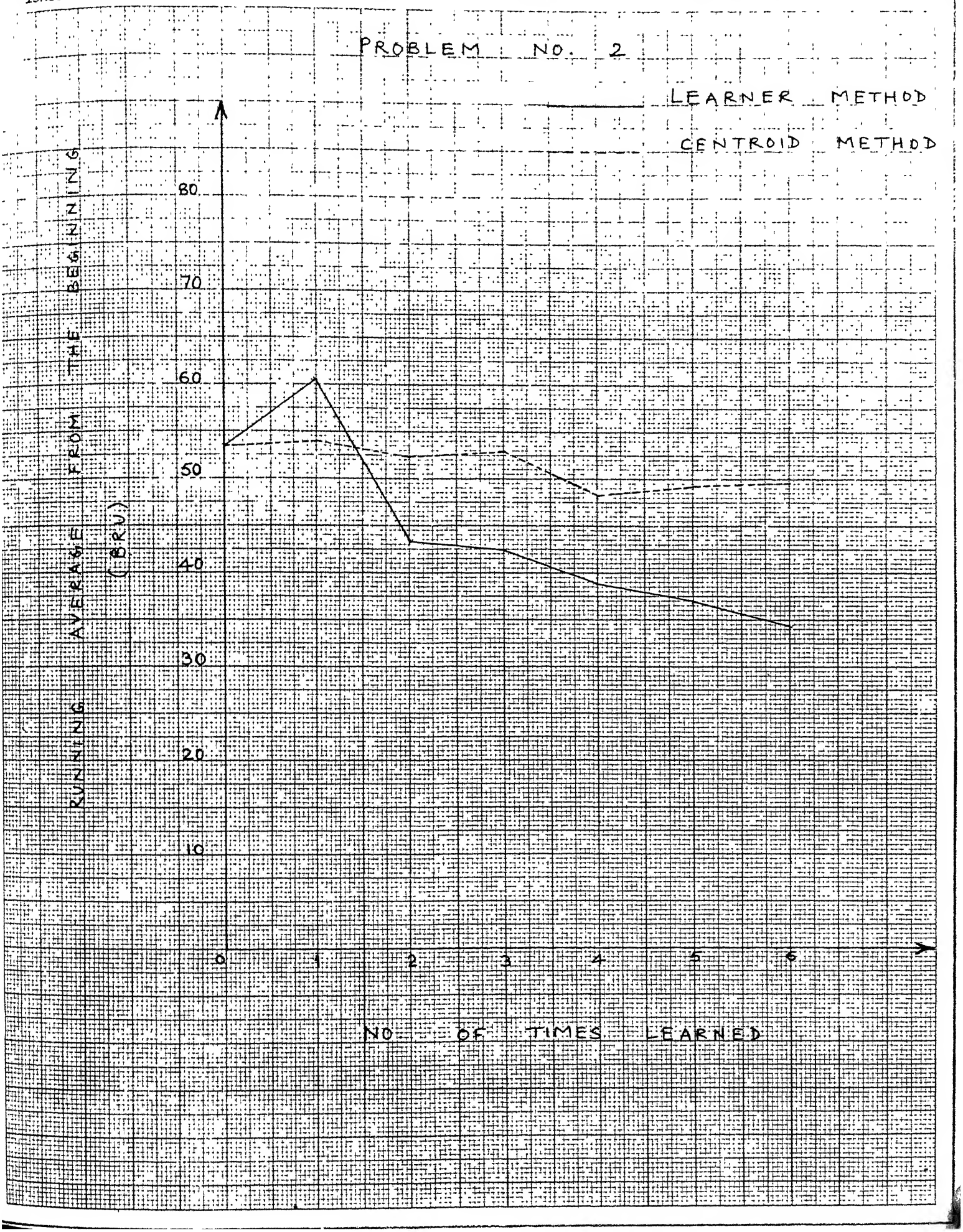


PROBLEM NO. 2

LEARNER METHOD

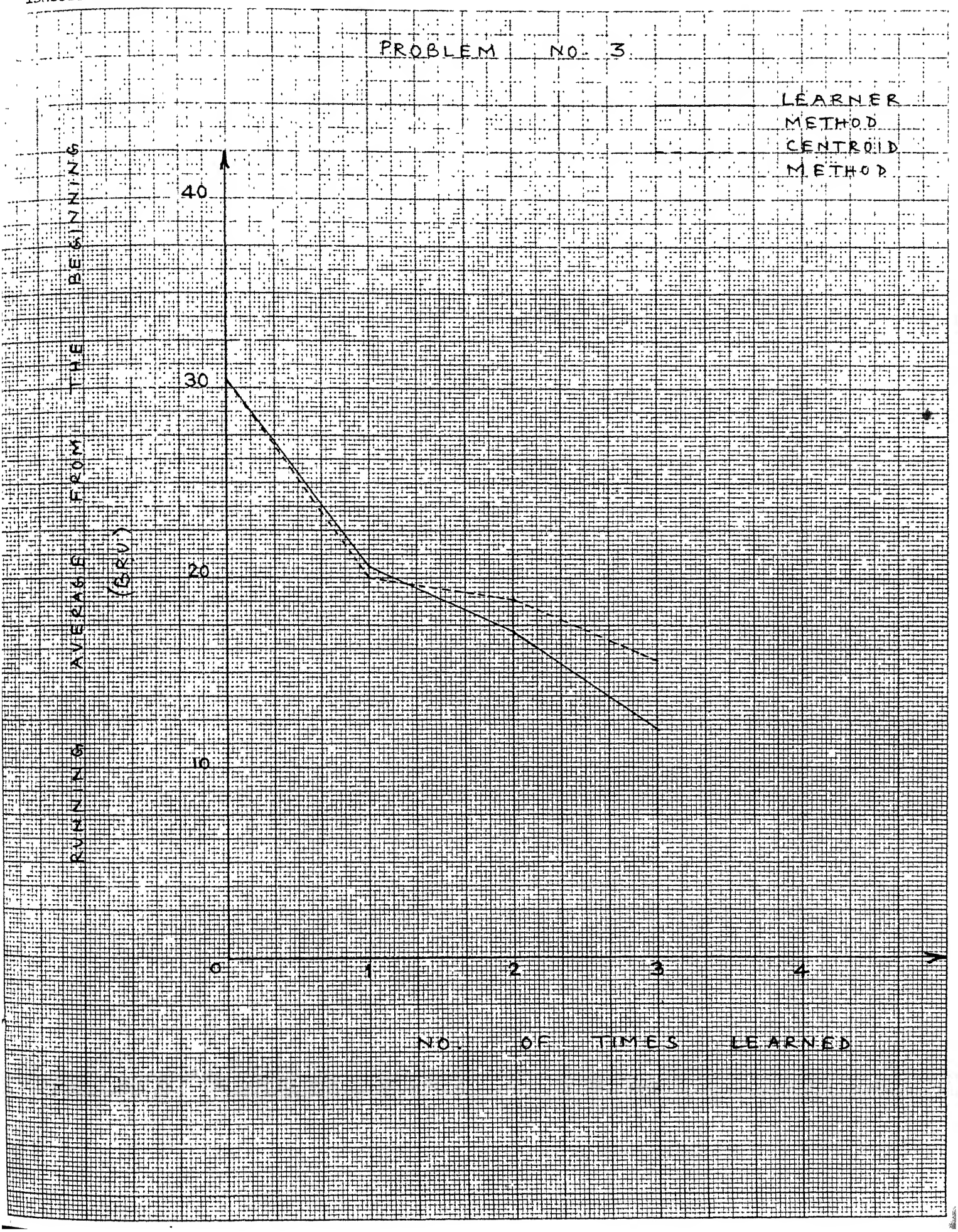
CENTROID METHOD

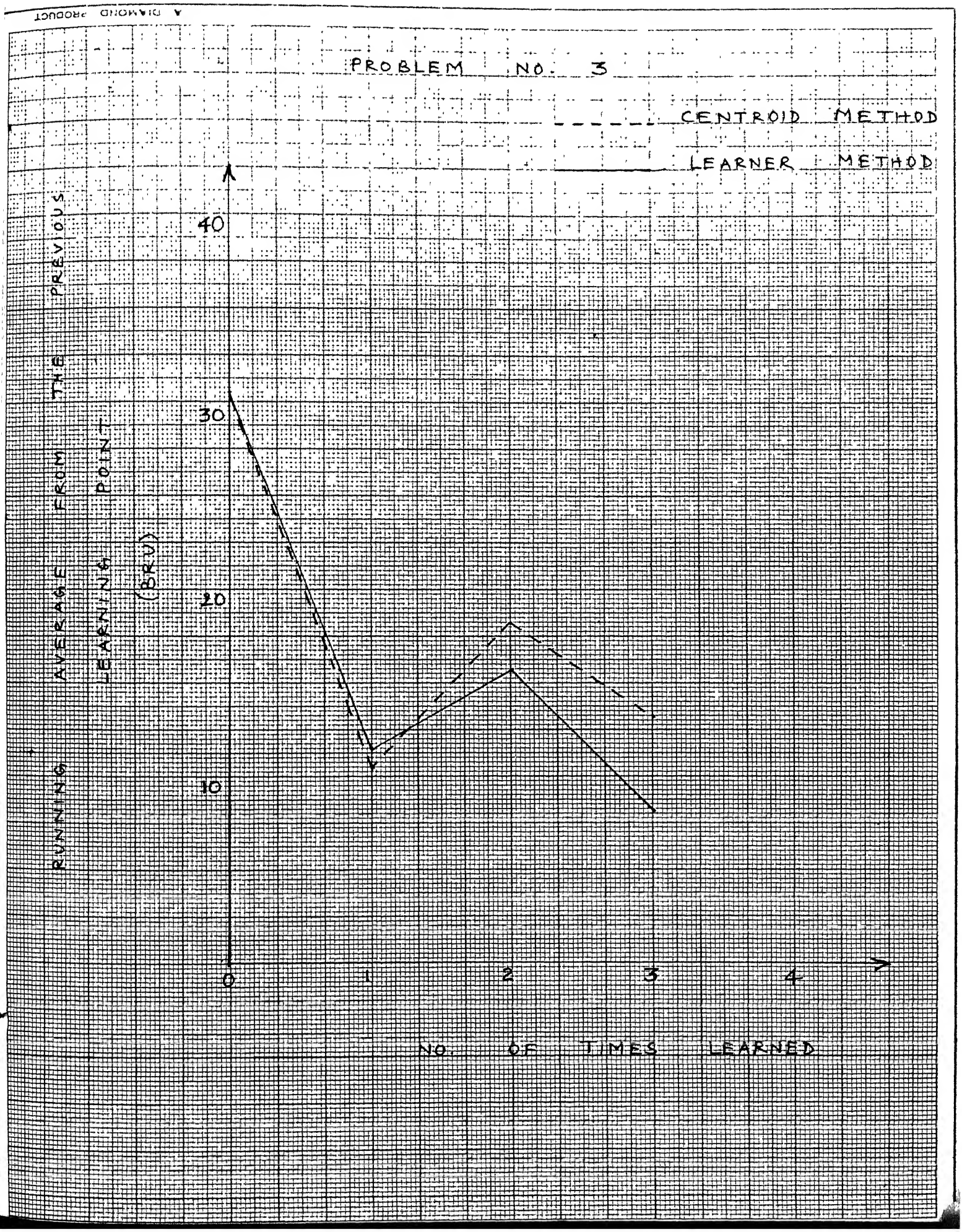


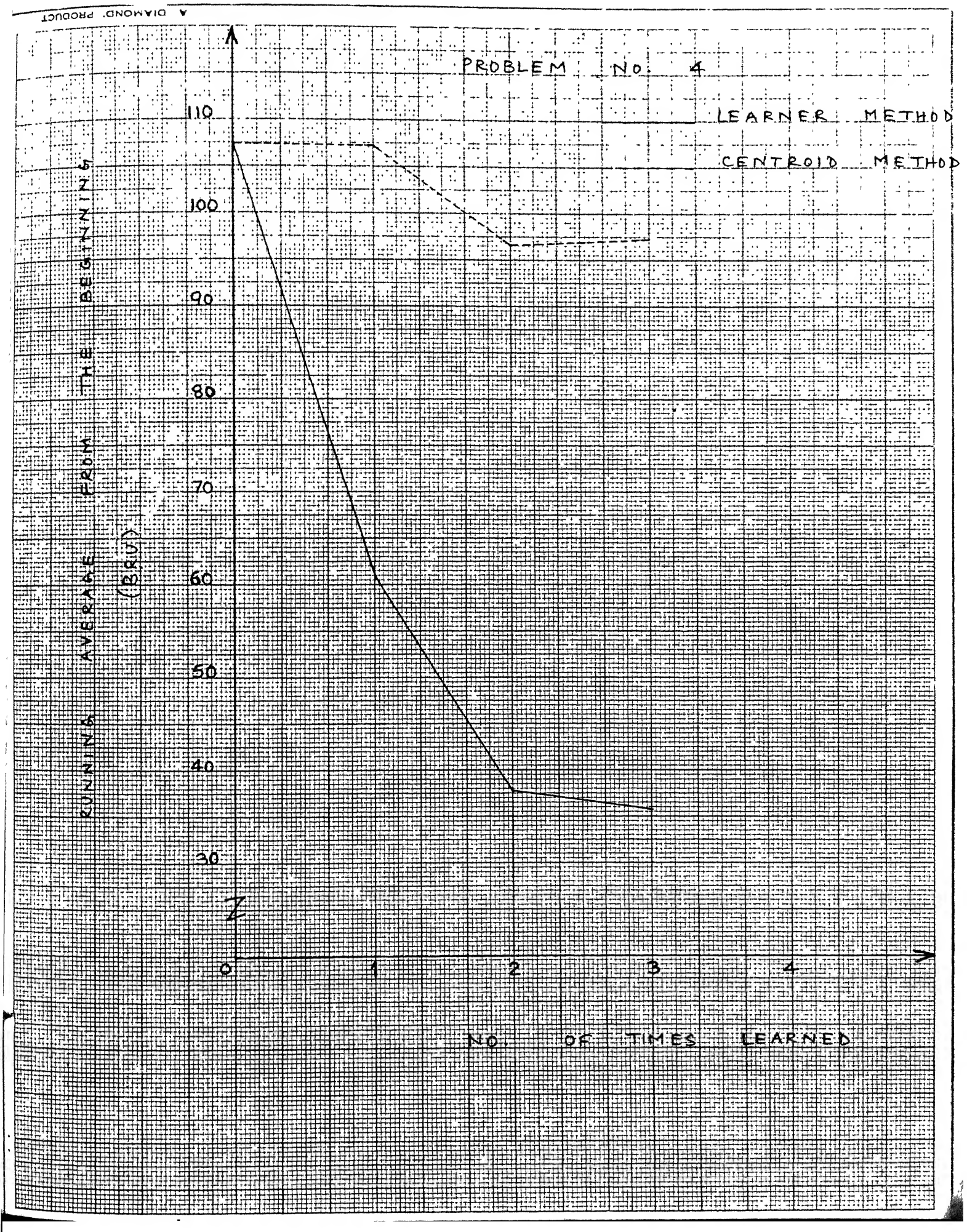


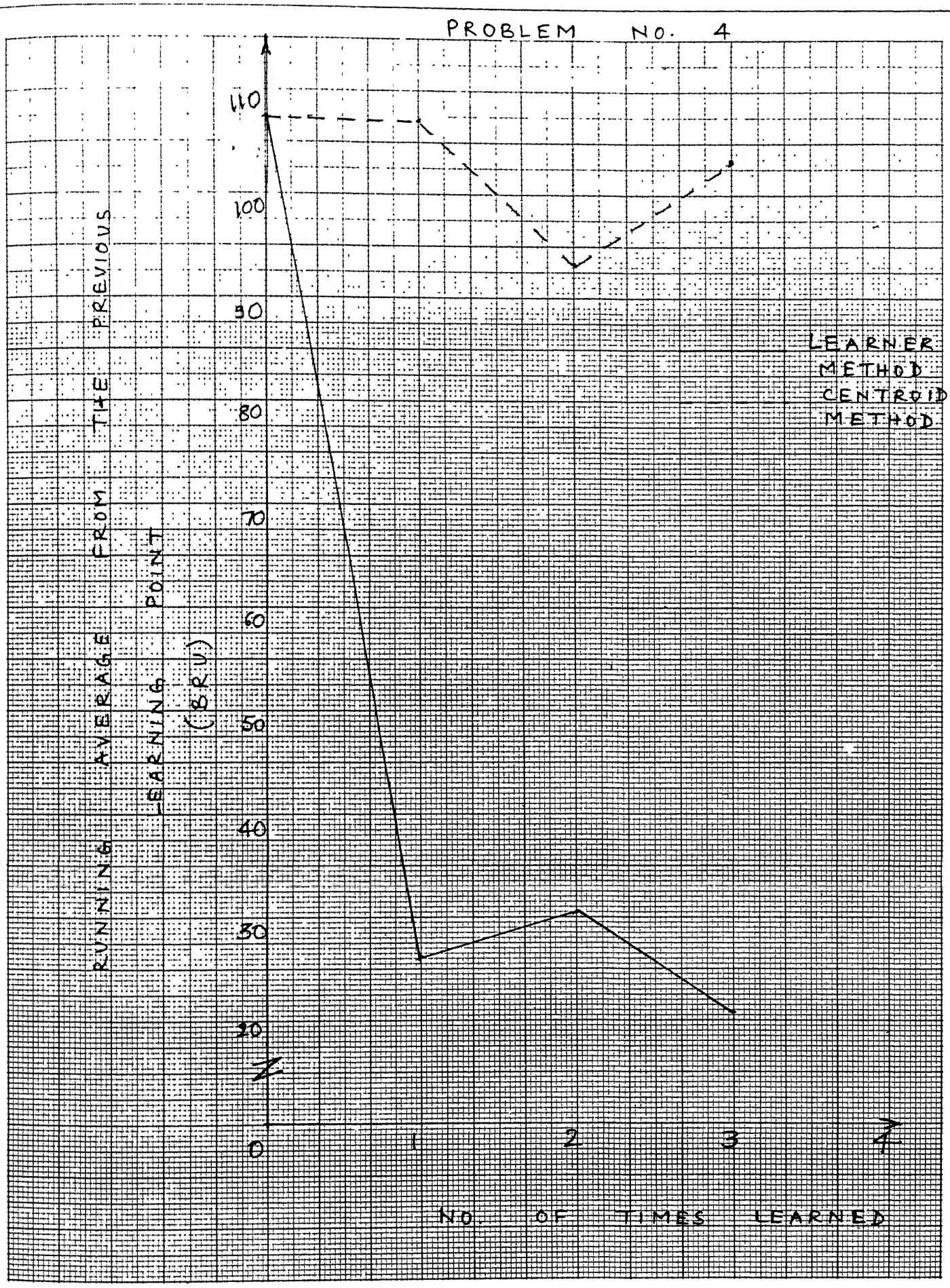
PROBLEM NO. 3

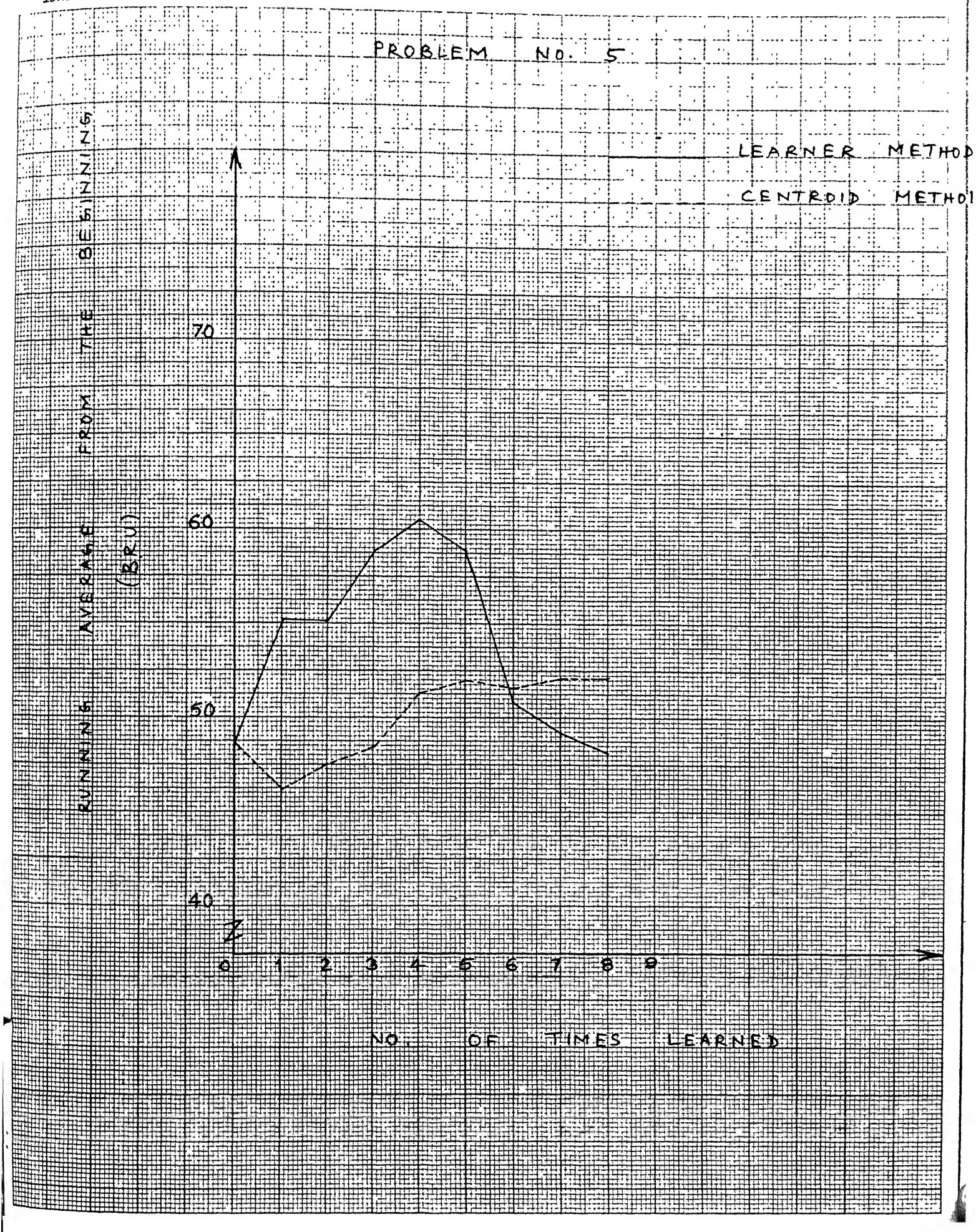
LEARNER METHOD
CENTROID METHOD

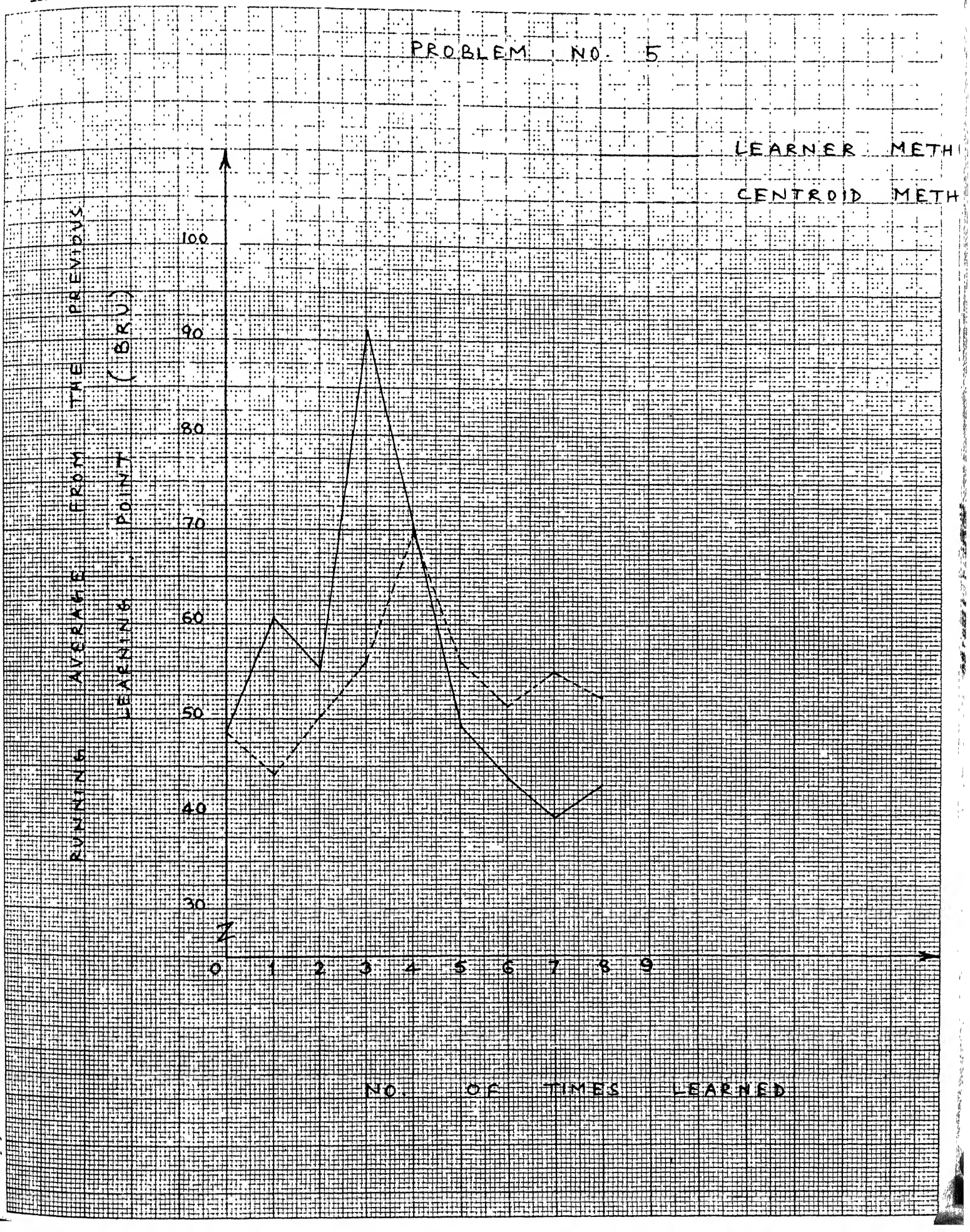


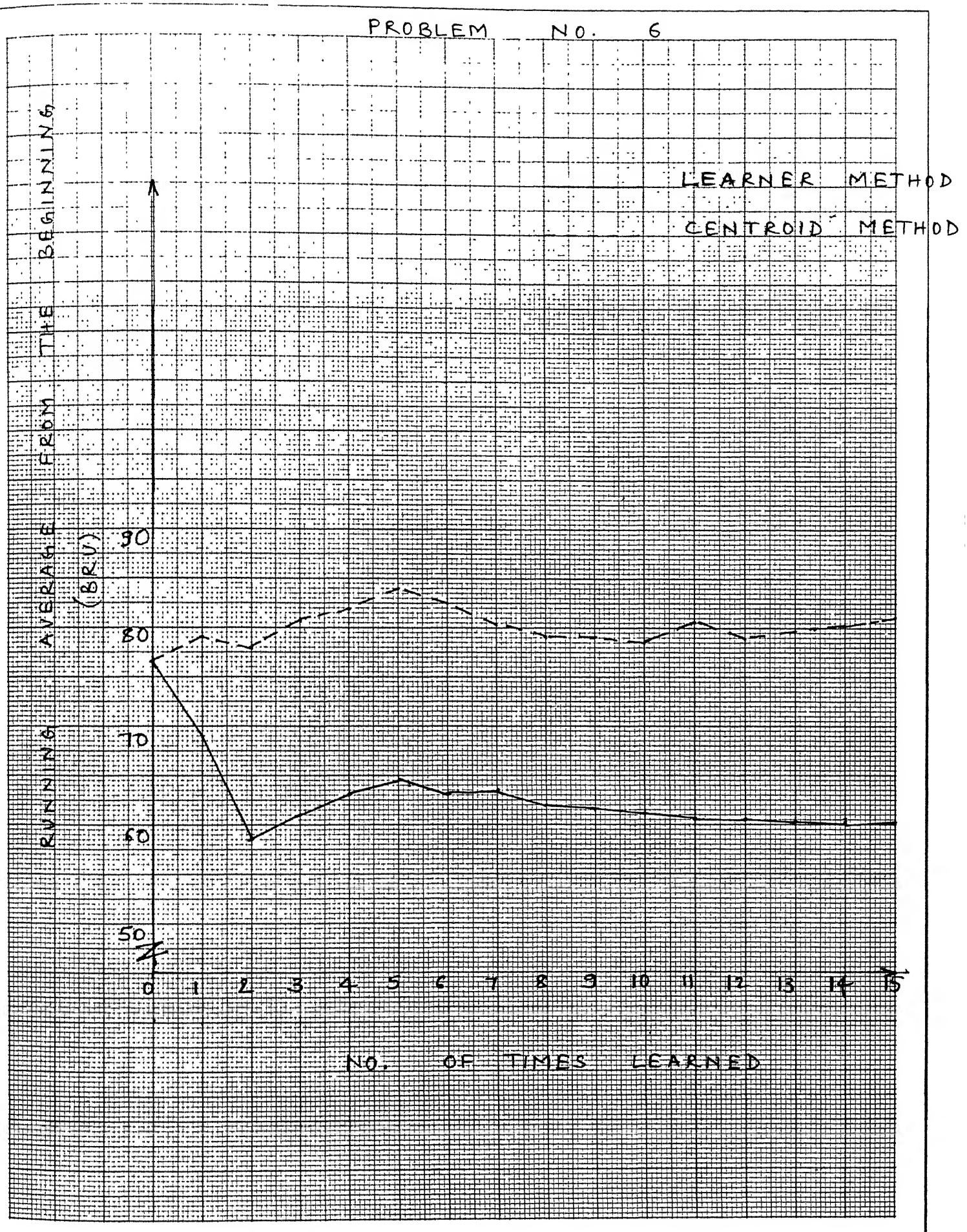

PROBLEM NO. 3
CENTROID METHOD
LEARNER METHOD
RUNNING AVERAGE FROM THE PREVIOUS LEARNING POINT (BRUD)
40
30
20
10
0
1
2
3
4
NO. OF TIMES LEARNED

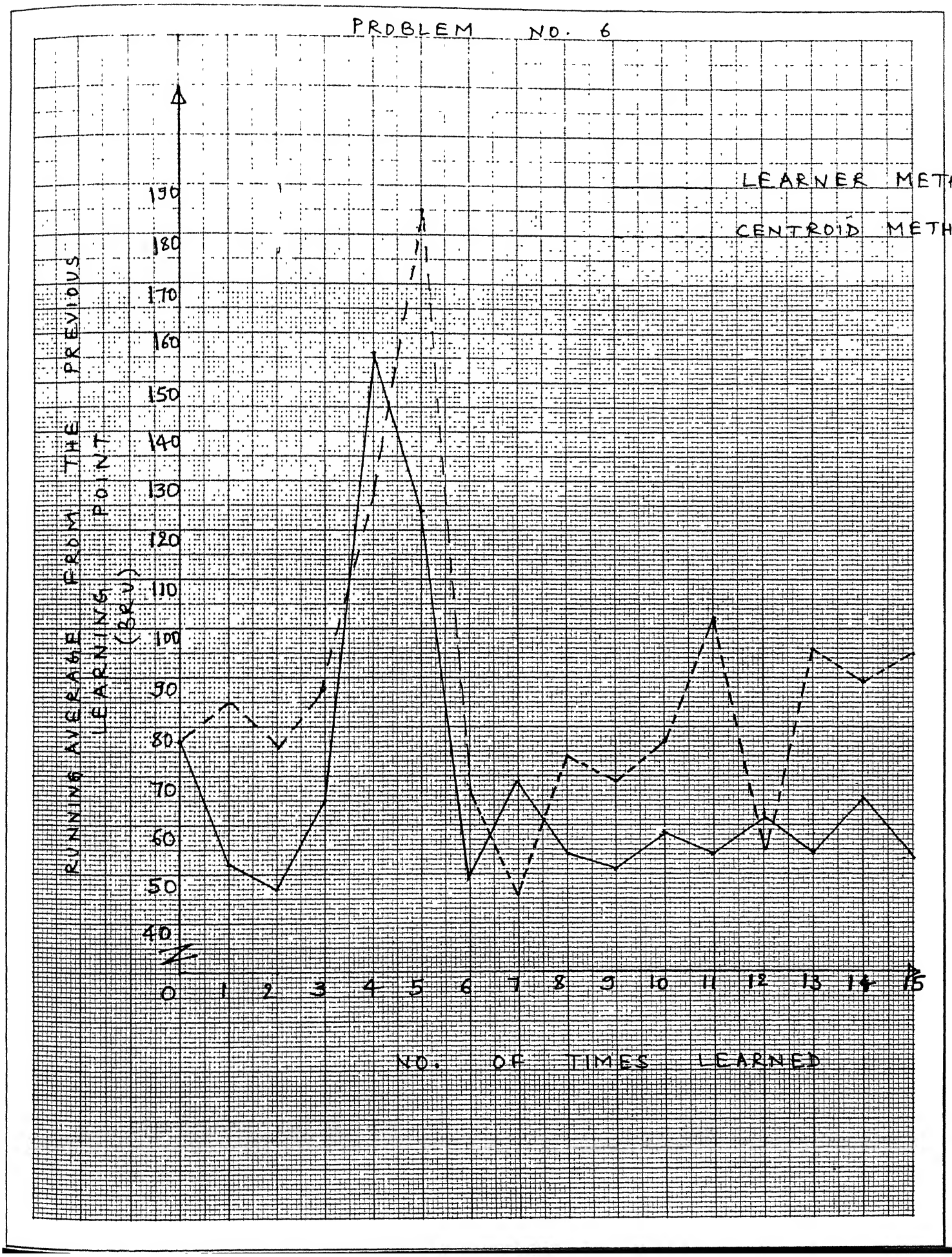

PROBLEM No. 4
LEARNER METHOD
CENTROID METHOD
AVERAGE ERROR (RMS) THE BEGINNERS RUNNING
110
100
90
80
70
60
50
40
30
0
1
2
3
4
NO. OF TIMES LEARNED








PROBLEM NO. 6
LEARNER METHOD
CENTROID METHOD
AVERAGE RUNNING FROM THE BEGINNING (BRV)
90
80
70
60
50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
NO. OF TIMES LEARNED


PROBLEM NO. 6
LEARNER METHO
CENTROID METHOD
RUNNING AVERAGE FROM THE PREVIOUS LEARNINGS POINT (BRU)
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
NO. OF TIMES LEARNED

distributions where there exists a centroid for the "determining distribution" and the distribution decreases radially outwards , Explorer has performed slightly better than the centroid method (problems 3 and 5). This is very creditable for the Explorer because the distribution generated was expected to be favourable to the centroid method . In fact for such distributions the centroid method has to theoretically work better or same as any other strategy . Perhaps , because the generated distribution is not the same as the ideal distribution that was expected to be generated , the Explorer was able to take an edge over the centroid method .

For other distributions the performance of Explorer was excellent compared to the centroid method . In some of our experiments a mixture of 2 distributions was used. In the industrial situation the actual set being assembled by the robot can be from two different samples (and it is quite likely that these distributions will be different) . Problem number 4 is indicative of how much one can stand to gain by the use of the learner method . The Explorer has on an average beaten the centroid method by a distance of more than 6Ø BRU for each search and if we consider only the last lap then the centroid method is being beaten by a distance of more than 8Ø BRUs for each search .

It has also been seen that the Explorer found it tougher to learn the normal distributions as compared to other distributions that were fed in . However , since finally the Explorer is able to perform as good as the centroid method we can well assume that it has learnt the normal distribution properly (this is a valid claim since no strategy can be theoretically expected to perform better

than the centroid method for a normal bivariate distribution).

Let us perform a small calculation to see what can theoretically be the best search average for problem number 1 .

Since , the points are uniformly distributed over an area varying from -5 to +5 in the x axis , and 4 to 6 in the y axis , the best strategy on an average must take (11 * 3) /2 = 16.5 BRU.

In the last lap our algorithm seems to have performed better than the best strategy ! Actually this is true only if it is assumed that the simulation has generated a perfect distribution . It was found that the simulation technique had generated lesser number of points at the periphery than in the interior region of the distribution . Thus we must actually make an adjustment for this while making the calculation .

4. Conclusions :

The Explorer has clearly proved to be superior to the centroid method. It has also shown an excellent ability to be able to learn the distributions with which the hole keeps appearing . Throughout our implementation we have used the decision making approach to develop the learner . Most of the heuristics used are very simple mathematical formulae . The motivation behind the choice of these formulae has been to simulate the human thinking . Though most of the formulae are only crude representation of the way human beings make decisions , these formulae have proved to be quite useful in

our implementation . We believe that most of the power in our system was derived because of our approach to look at the whole problem from the point of view of decision making.

APPENDIX A

This appendix gives the complete statistics for the six problems with which the experiments were performed.

Sample run on the computer.

```
Problem no. : 1

Data fed in :

No of points = 200

Distribution in X direction : uniform -5 to +5

Distribution in Y direction : uniform +4 to -7

Mean and Variance in the x and y directions

expected value in x = nil

expected value in y = nil

variance in x =       nil

variance in y =       nil

**transformation applied :

*center :             nil

*alpha :              nil

no of samples =        1

mix / no-mix =        nil
```

**The transformation applied is

1) rotate about z axis by *alpha

2) followed by shift of z axis to *center.

If option for two samples is given then the two set of points (the transformed and the untransformed points) are generated. Otherwise only the transformed points are taken into account. The two samples are mixed if the mix option is given .

Statistics for the sample run :

The table below gives the progressive statistical analysis of the sample run .Each block shows the statistics for the two programs . A block is shown after each learning .The figures given are as follows :

column 1 : shows the learning point from which onwards the statistics is calculated in the particular row < lno > .

column 2 : average distance travelled from lno to current learning point by the PLS method < av-PLS >.

column 3 : average distance travelled from lno to the current learning point by the centroid method < av-CEN> .

column 4 : variance by the PLS method < var-PLS >.

column 5 : variance by the centroid method < var-cent >.

In each block the current learning point is indicated as < current learning point > at < prob-no. > .

1 at 12

	av-PLS	av-CEN	var-PLS	var-cent
Ø :	33.58	33.58	948.91	948.91

---

2 at 17

Ø :	42.12	34.65	97Ø.1Ø	888.46

3 :	14.99	38.96	26Ø.42	1188.83
4 :	13.35	41.12	83.66	1297.76

Sample run on the computer.

```
Problem no. : 2

Data fed in :

No of points =  3ØØ

Distribution in X direction :  uniform

Distribution in Y direction :  uniform

Mean and Variance in the x and y directions

expected value in x =    nil

expected value in y =    nil

variance in x =          nil

variance in y =          nil

**transformation applied :

*center :          (Ø Ø)

*alpha :           3.14

no of samples =      2

mix / no-mix =       mix
```

	av-PLS	av-LEN	var-PLS	var-cent
		1 at 26		
Ø :	53.46	53.46	1511.56	1511.56
		2 at 38		
Ø :	6Ø.5Ø	53.95	2555.36	1289.Ø5
1 :	75.75	55.ØØ	4477.Ø2	8Ø5.33
		3 at 72		
Ø :	43.22	52.39	1942.34	1Ø41.6Ø
1 :	37.43	51.78	2Ø93.Ø7	774.95
2 :	23.91	5Ø.65	55Ø.67	759.29
		4 at 1Ø4		
Ø :	42.39	52.97	1915.49	1Ø4Ø.64
1 :	38.71	52.81	1995.7Ø	883.57
2 :	31.97	52.41	1249.67	896.76
3 :	4Ø.53	54.28	185Ø.Ø6	1Ø36.Ø1

---

5 at 176

Ø :	38.86	48.23	1464.77	954.98
1 :	36.33	47.32	1413.29	852.94
2 :	32.9Ø	46.65	999.98	851.5Ø
3 :	35.84	45.35	1111.83	874.73
4 :	33.75	41.38	769.58	751.79

---

6 at 23Ø

Ø :	36.97	49.3Ø	1223.82	966.76
1 :	34.86	48.76	1148.Ø4	894.83
2 :	32.31	48.38	828.96	897.84
3 :	34.11	47.89	87Ø.42	926.3Ø
4 :	32.48	46.26	6Ø8.5Ø	885.42
5 :	3Ø.8Ø	52.78	388.75	989.28

Number of problems solved : 3ØØ

Ø :	34.36	49.6Ø	1Ø87.72	1Ø1Ø.14
1 :	32.54	49.23	1ØØ9.58	961.Ø1
2 :	3Ø.56	48.97	761.35	966.55
3 :	31.56	48.72	785.18	996.97
4 :	3Ø.Ø9	47.81	596.Ø2	984.72
5 :	27.97	51.55	482.97	1Ø81.96
6 :	25.79	5Ø.6Ø	544.71	1151.38

Sample run on the computer.

Problem no. : 3

Data fed in :

No of points = 2ØØ

Distribution in X direction : exponential

Distribution in Y direction : exponential

Mean and Variance in the x and y directions

expected value in x = 2

expected value in y = 2

variance in x = nil

variance in y = nil

**transformation applied :

*center : nil

*alpha : nil

no of samples = 1

mix / no-mix = nil

	av-PLS	av-LEN	var-PLS	var-cent
		1 at 12		
Ø :	3Ø.42	3Ø.42	43Ø4.Ø8	43Ø4.Ø8
		2 at 25		
Ø :	2Ø.6Ø	2Ø.ØØ	2164.96	2227.52
1 :	11.54	1Ø.38	19.33	118.Ø8
		3 at 84		
Ø :	17.15	18.92	1575.8Ø	1476.6Ø
1 :	14.94	17.ØØ	1Ø86.89	979.64
2 :	15.69	18.46	1318.99	1157.71

Number of problems solved : 2ØØ

Ø :	11.98	15.73	711.46	1ØØ8.47
1 :	1Ø.81	14.79	459.Ø8	783.45
2 :	1Ø.75	15.11	491.71	831.34

3 :	8.24	13.41	52.2Ø	656.76

Sample run on the computer.

Problem no. : 4

Data fed in :

No of points = 2ØØ

Distribution in X direction : exponential

Distribution in Y direction : exponential

Mean and Variance in the x and y directions

expected value in x = 2

expected value in y = 2

variance in x = nil

variance in y = nil

**transformation applied :

*center : (5 Ø)

*alpha : 3.14

no of samples = 2

mix / no-mix = mix

	av-PLS	av-LEN	var-PLS	var-cent
		1 at 18		
Ø :	1Ø7.5Ø	1Ø7.5Ø	2719.81	2719.81
		2 at 42		
Ø :	61.12	1Ø7.36	3Ø2Ø.15	1672.94
1 :	26.33	1Ø7.25	421.97	887.77
		3 at 175		
Ø :	37.82	96.61	2192.ØØ	896.Ø7
1 :	29.83	95.36	1511.Ø6	671.81
2 :	3Ø.47	93.21	17Ø4.97	6Ø2.71

Number of problems solved : 2ØØ

Ø :	35.68	97.44	2Ø47.66	892.29
1 :	28.58	96.45	142Ø.58	7ØØ.54
2 :	28.92	94.8Ø	1571.39	651.68

3 :	2Ø.68	1Ø3.28	78Ø.14	826.84

Sample run on the computer.

```
Problem no. : 5

Data fed in :

No of points =  239

Distribution in X direction :  normal

Distribution in Y direction :  normal

Mean and Variance in the x and y directions

expected value in x =   Ø

expected value in y =   Ø

variance in x =         3

variance in y =         3

**transformation applied :

*center :       nil

*alpha :        nil

no of samples =   1

mix / no-mix =   nil
```

	av-PLS	av-LEN	var-PLS	var-cent
		1 at 16		
Ø :	48.63	48.63	1126.61	1126.61
		2 at 35		
Ø :	55.17	46.17	1653.34	1Ø44.2Ø
1 :	6Ø.68	44.11	2Ø3Ø.43	965.46
		3 at 52		
Ø :	55.23	47.54	1417.1Ø	1ØØ8.33
1 :	58.17	47.Ø6	1518.19	955.ØØ
2 :	55.35	5Ø.35	93Ø.7Ø	922.7Ø
		4 at 58		
Ø :	58.93	48.43	1776.82	984.45
1 :	62.86	48.36	1968.65	93Ø.28
2 :	64.65	51.87	191Ø.49	873.94
3 :	91.ØØ	56.17	3747.33	71Ø.81

---

5 at 67

Ø :	6Ø.49	51.31	1713.38	1Ø9Ø.81
1 :	64.22	52.16	1839.42	1Ø76.6Ø
2 :	66.31	56.94	1714.21	1Ø81.25
3 :	78.73	64.4Ø	2311.8Ø	1156.11
4 :	7Ø.56	69.89	1187.58	1377.65

---

6 at 78

Ø :	58.91	51.99	18ØØ.52	1121.58
1 :	61.56	52.85	194Ø.Ø8	1116.61
2 :	61.95	56.72	1899.67	1134.62
3 :	66.27	6Ø.88	2486.12	1229.33
4 :	58.85	62.3Ø	1869.23	1376.21
5 :	49.27	56.Ø9	2223.11	1289.36

---

7 at 167

Ø :	5Ø.81	51.6Ø	14Ø2.58	1172.2Ø
1 :	51.Ø5	51.91	1431.26	1176.ØØ
2 :	49.66	53.Ø4	1329.72	1196.26
3 :	48.82	53.43	1383.21	1235.48
4 :	46.5Ø	53.28	1149.74	1263.93
5 :	44.33	51.79	1Ø89.54	1226.65
6 :	43.72	51.26	946.Ø4	1216.33

---

8 at 195

Ø :	49.2Ø	52.Ø9	1377.5Ø	12ØØ.99
1 :	49.25	52.4Ø	1399.9Ø	12Ø6.46
2 :	47.89	53.39	13Ø7.66	1225.94
3 :	47.Ø1	53.75	1345.Ø7	126Ø.76
4 :	45.Ø8	53.64	1151.39	1284.58
5 :	43.29	52.5Ø	11ØØ.ØØ	1258.17
6 :	42.73	52.16	99Ø.73	1253.91
7 :	39.57	55.Ø4	1119.67	1362.53

Number of problems solved : 239

Ø :	48.Ø8	52.14	1353.91	1192.46
1 :	48.Ø4	52.39	137Ø.19	1196.24
2 :	46.86	53.16	1292.42	121Ø.74
3 :	46.Ø9	53.42	1318.16	1236.15
4 :	44.6Ø	53.33	1168.56	1253.3Ø
5 :	43.24	52.46	113Ø.48	1231.69
6 :	42.83	52.21	1Ø53.17	1226.79
7 :	41.74	53.39	1183.42	1237.21
8 :	43.11	52.34	1219.1Ø	1154.63

Sample run on the computer.

Problem no. : 6

Data fed in :

No of points = 398

Distribution in X direction : normal

Distribution in Y direction : normal

Mean and Variance in the x and y directions

expected value in x = 4

expected value in y = Ø

variance in x = 2

variance in y = 2

**transformation applied :

*center : (Ø Ø)

*alpha : 3.14

no of samples = 2

mix / no-mix = mix

	av-PLS	av-LEN	var-PLS	var-cent
		1 at 43		
Ø :	76.7Ø	76.7Ø	5735.33	5735.33
		2 at 63		
Ø :	68.79	79.22	5333.47	4743.82
1 :	51.8Ø	84.65	4Ø46.36	2568.93
		3 at 115		
Ø :	59.Ø5	77.83	4213.55	3597.Ø4
1 :	48.51	78.51	3Ø07.69	2318.78
2 :	47.25	76.15	26Ø2.46	22Ø2.51
		4 at 166		
Ø :	6Ø.96	8Ø.83	4Ø02.37	4257.36
1 :	55.46	82.27	3279.74	3732.64
2 :	56.17	81.81	3127.78	3957.28

3 :	65.27	87.57	3499.38	5680.66

5 at 170

Ø :	63.19	81.94	4337.99	4253.11
1 :	58.61	83.72	3782.16	3738.80
2 :	59.89	83.54	3722.47	3957.28
3 :	71.84	90.53	4487.63	5515.92
4 :	155.50	128.25	9539.25	1880.69

6 at 174

Ø :	64.58	84.32	4325.83	4560.79
1 :	60.60	86.82	3799.16	4149.95
2 :	62.19	87.21	3738.14	4433.82
3 :	75.36	96.95	4369.01	6197.81
4 :	139.63	156.75	5134.48	5357.44
5 :	123.75	185.25	225.69	7209.69

7 at 192

Ø :	63.24	82.66	4217.96	4358.00
1 :	59.36	84.38	3712.71	3947.31
2 :	60.53	84.33	3650.76	4161.00
3 :	69.49	89.86	4159.26	5407.91
4 :	77.77	94.35	5350.25	4842.46
5 :	63.64	88.18	3290.32	5133.97
6 :	50.28	66.61	2989.87	2113.57

8 at 205

Ø :	63.58	80.34	4209.59	4216.18
1 :	60.10	81.30	3746.82	3808.51
2 :	61.27	80.83	3693.56	3981.29
3 :	69.37	83.53	4144.85	4989.09
4 :	74.72	78.26	4938.41	4035.57
5 :	65.49	72.54	3581.56	3963.56
6 :	57.97	58.00	3520.03	1694.13
7 :	68.62	46.08	4058.85	868.53

9 at 247

Ø :	61.88	79.32	3936.98	4054.00
1 :	58.75	79.87	3501.87	3697.85
2 :	59.51	79.35	3436.86	3817.80
3 :	64.34	80.61	3683.01	4448.51
4 :	63.75	76.23	3797.74	3623.07
5 :	58.99	73.53	3039.49	3565.73
6 :	55.44	67.41	2951.26	2644.71
7 :	57.13	67.67	2927.06	2818.26
8 :	53.57	74.36	2523.25	3232.71

10 at 255

Ø :	61.53	79.ØØ	3844.81	4Ø45.ØØ
1 :	58.45	79.47	34Ø5.23	37ØØ.86
2 :	59.15	78.93	3333.35	3815.68
3 :	63.56	79.96	3532.75	441Ø.94
4 :	62.58	75.6Ø	3549.23	3631.12
5 :	58.21	73.12	2841.96	3576.88
6 :	54.98	67.58	2748.57	2745.9Ø
7 :	56.32	67.86	2671.52	2926.22
8 :	53.12	73.52	2261.27	33Ø5.81
9 :	5Ø.75	69.13	879.19	3666.61

11 at 29Ø

Ø :	61.Ø5	78.72	38Ø1.38	4163.73
1 :	58.32	79.Ø7	3414.64	3889.29
2 :	58.9Ø	78.58	3354.9Ø	4ØØ2.64
3 :	62.36	79.3Ø	3526.18	4535.27
4 :	61.16	75.9Ø	3532.28	4Ø24.46
5 :	58.Ø2	74.15	3Ø25.5Ø	4ØØ1.51
6 :	55.75	7Ø.32	2967.91	345Ø.58
7 :	56.76	71.ØØ	2957.37	3693.16
8 :	54.94	74.81	2764.1Ø	4Ø15.64
9 :	56.28	75.26	2995.74	4779.96
1Ø :	57.54	76.66	347Ø.93	5Ø23.88

12 at 312

Ø :	6Ø.55	8Ø.38	3728.88	42Ø4.36
1 :	57.97	8Ø.97	3359.81	3957.11
2 :	58.47	8Ø.68	33Ø1.36	4Ø67.44
3 :	61.43	81.87	3443.87	4552.87
4 :	6Ø.Ø8	79.88	3417.5Ø	4143.62
5 :	57.39	78.52	2981.37	4139.62
6 :	55.47	75.43	2929.92	371Ø.88
7 :	56.25	76.75	2916.27	3937.Ø7
8 :	54.75	8Ø.48	2756.62	4181.69
9 :	55.51	84.43	29Ø5.94	4755.Ø4
1Ø :	56.18	86.58	3186.78	487Ø.31
11 :	54.ØØ	1Ø2.36	2727.ØØ	422Ø.23

13 at 333

Ø :	6Ø.58	78.81	3583.87	4317.Ø2
1 :	58.19	79.13	322Ø.63	41Ø5.96
2 :	58.66	78.72	3156.22	4217.39
3 :	61.39	79.33	3249.82	4696.Ø6
4 :	6Ø.2Ø	76.81	3167.58	4368.31
5 :	57.86	75.55	2782.87	4362.84
6 :	56.2Ø	72.79	2735.23	398Ø.87
7 :	56.96	73.58	2697.67	4213.75
8 :	55.77	76.38	2544.22	4468.86
9 :	56.85	77.36	255Ø.94	5Ø69.6Ø
1Ø :	57.47	78.21	2718.2Ø	52Ø5.83
11 :	57.42	79.47	21Ø5.5Ø	535Ø.39
12 :	61.ØØ	55.48	1429.33	54Ø9.58

14 at 351

Ø :	6Ø.26	79.7Ø	35Ø7.66	4263.88
1 :	57.97	8Ø.12	3153.69	4Ø57.Ø1
2 :	58.4Ø	79.81	3Ø88.87	4158.83
3 :	6Ø.86	8Ø.61	3162.62	4586.29
4 :	59.64	78.7Ø	3Ø62.92	4267.58
5 :	57.52	77.6Ø	2712.23	4264.87
6 :	56.Ø2	75.17	2667.Ø5	3930.52
7 :	56.67	76.14	2626.35	4126.99
8 :	55.61	78.82	2484.96	4329.49
9 :	56.43	8Ø.62	2467.15	4761.16
1Ø :	56.91	81.57	2596.56	484Ø.45
11 :	56.54	84.39	2Ø94.51	4713.39
12 :	57.97	74.26	1732.Ø2	47Ø6.65
13 :	54.44	96.17	2Ø62.Ø2	2995.Ø3

---

15 at 375

Ø :	6Ø.6Ø	8Ø.27	3443.23	4215.42
1 :	58.51	8Ø.73	31Ø8.44	4Ø16.7Ø
2 :	58.94	8Ø.48	3Ø45.25	41Ø8.46
3 :	61.28	81.34	31ØØ.99	4485.16
4 :	6Ø.31	79.82	2998.94	4181.67
5 :	58.45	78.88	2691.Ø6	4179.91
6 :	57.15	76.76	2653.57	3889.96
7 :	57.83	77.76	2615.39	4Ø53.56
8 :	57.ØØ	8Ø.18	2495.42	4214.49
9 :	58.13	82.Ø9	2481.17	4521.85
1Ø :	58.62	82.96	2584.1Ø	4566.91
11 :	59.Ø6	85.55	2218.27	4355.66
12 :	6Ø.83	79.68	2Ø28.56	4269.8Ø
13 :	6Ø.74	91.79	2328.15	326Ø.45
14 :	65.46	88.5Ø	2475.75	3434.33

Number of problems solved : 398

Ø :	6Ø.17	81.1Ø	3413.23	4297.67
1 :	58.17	81.64	3Ø94.85	412Ø.9Ø
2 :	58.55	81.46	3Ø35.48	4212.98
3 :	6Ø.62	82.43	3Ø87.3Ø	4576.27
4 :	59.6Ø	81.3Ø	299Ø.9Ø	4326.42
5 :	57.92	8Ø.48	2711.84	4329.98
6 :	56.74	78.61	2677.46	4Ø79.Ø3
7 :	57.31	79.66	2646.19	4237.1Ø
8 :	56.54	81.92	2541.84	4382.94
9 :	57.37	84.Ø2	2543.88	4682.55
1Ø :	57.74	84.85	2634.42	4726.28
11 :	57.81	87.51	2363.3Ø	46Ø1.Ø1
12 :	58.78	83.71	2265.61	4627.53
13 :	58.Ø6	92.83	2533.69	4Ø34.14
14 :	59.45	91.55	27Ø7.4Ø	4426.2Ø
15 :	53.17	94.74	2872.Ø6	5441.32

Sample run on the computer.

```
Data for in :

No of points = 200

Distribution in X direction : exponential

Distribution in Y direction : exponential

Mean and Variance in the x and y directions

expected value in x =  2

expected value in y = 2

variance in x =          nil

variance in y =          nil

**transformation applied :

*center :          (5 0)

*alpha :           3.14

no of samples =    2

*mix =             mix
```

**The transformation applied is

1) rotate about x axis by *alpha

2) followed by shift of x axis to *center.

If option for two samples is given then the two set of points (the transformed and the untransformed points) are generated. Otherwise only the transformed points are taken into account. The two samples are mixed if the mix option is given .

The plot is of the distance travelled vs the problem no. On the x-axis the problem number beins solved by the robot is plotted and on the y-axis the distance travelled by it to find the hole .

Note : on the y axis 100 units represents a distance of 300 BRU .

* is used for the centroid method (plotted in red ink)

is used for the PLS method (plotted in green ink)

Statistics for the sample run :

The table below gives the progressive statistical analysis of the sample run .Each block shows the statistics for the two programs . A block is shown after each learning .The figures given are as follows :

column 1 : shows the learning point from which onwards the statistics is calculated in the subsequent columns < lno > .

column 2 : average distance travelled from lno to current learning point by the PLS method < av-PLS >.

column 3 : average distance travelled from lno to the current learning point by the centroid method < av-cen> .

column 4 : square of the variance by the PLS method < var-PLS >.

column 5 : square of the variance by the centroid method < var-cen >.

In each block the current learning point is indicated as < current learning point > at < prob-no. > .

A pointer >> is put on the x-axis to indicate the points at which the program had learnt .

		1 at 18		
0 :	107.50	107.50	2719.81	2719.81
		2 at 42		
0 :	61.12	107.36	3020.15	1672.94
1 :	26.33	107.25	421.97	887.77
		3 at 175		
0 :	37.82	96.61	2192.00	896.07
1 :	29.83	95.36	1511.06	671.81
2 :	30.47	93.21	1704.97	602.71

Number of problems solved : 200

0 :	35.68	97.44	2047.66	892.29
1 :	28.58	96.45	1420.58	700.54
2 :	28.92	94.80	1571.39	651.68
3 :	20.68	103.28	780.14	826.84